

**AN INVESTIGATION OF AN AUTOMATIC MACHINE
GRADING SYSTEM FOR ASSEMBLY LANGUAGE INSTRUCTION**

by

Roger Francis Bacon

United States Naval Postgraduate School



THESIS

AN INVESTIGATION OF
AN AUTOMATIC MACHINE GRADING SYSTEM
FOR ASSEMBLY LANGUAGE INSTRUCTION

by

Roger Francis Bacon

June 1969

This document has been approved for public release and sale; its distribution is unlimited.

LIBRARY
NAVAL POSTGRADUATE SCHOOL
MONTEREY, CALIF. 93940

An Investigation
of
An Automatic Machine Grading System
for
Assembly Language Instruction

by

Roger Francis Bacon
Lieutenant Commander, United States Navy
B.S., United States Naval Academy, 1959

Submitted in Partial Fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN COMPUTER SCIENCE

from the

NAVAL POSTGRADUATE SCHOOL
June 1969

3115
c.1

ABSTRACT

Machine grading techniques are becoming of greater importance because of the increased number of students in programming classes. Characteristics and limitations of automatic machine grading systems are proposed. A grader for introductory assembly language programming courses was developed. The properties of this grader are discussed and an example of a graded program is given.

TABLE OF CONTENTS

I. INTRODUCTION-----	7
A. BACKGROUND-----	7
B. OBJECTIVE-----	7
C. METHODOLOGY AND ORGANIZATION OF THESIS-----	8
II. CHARACTERISTICS OF A GRADER-----	9
A. REVIEW OF AUTOMATIC GRADING SYSTEMS-----	9
B. THE METRICS OF AN ALGORITHM-----	15
C. GRADER DESIGN CONSIDERATIONS-----	20
III. IMPLEMENTATION OF A GRADER-----	23
A. PURPOSE OF A GRADER-----	23
B. MECHANIZATION OF THE GRADER-----	24
C. USE OF THE GRADER-----	29
1. Student Grading Conventions-----	29
2. Instructor Grading Conventions-----	31
IV. CONCLUSIONS AND RECOMMENDATIONS-----	36
A. CONCLUSIONS-----	36
B. RECOMMENDATIONS-----	37
APPENDIX A THE STUDENT OPERATING SYSTEM-----	39
APPENDIX B THE GRADER PROGRAM-----	47
APPENDIX C FLOWCHART DOCUMENTATION OF THE GRADER PROGRAM-----	64
APPENDIX D EXAMPLES OF GRADED PROGRAMS-----	77
APPENDIX E EXAMPLE OF INSTRUCTOR'S ANSWER CONTROL SECTION---	97
BIBLIOGRAPHY-----	103
INITIAL DISTRIBUTION LIST-----	104
FORM DD 1473-----	105

LIST OF TABLES

Table		Page
I.	Summary of Grader Design Characteristics	22
II.	U. S. Naval Postgraduate School Computer Facility Usage Data	40

ACKNOWLEDGEMENTS

The author wishes to acknowledge the enthusiastic help given by G. L. Barksdale, Jr., of the Naval Postgraduate School. The author is also indebted to Professor Andries van Dam and Richard M. Kogut of Brown University for the opportunity to use the Brown University Student Operating System.

I. INTRODUCTION

A. BACKGROUND

The capability of an instructor to teach programming courses and to grade student programs in classes with large numbers of students has been significantly enhanced by automatic machine grading systems. The concept of computer-controlled grading of student programs was first documented by Hollingsworth [1] in 1960. Machine grading systems are known to exist at Rensselaer Polytechnic Institute [1], Stanford University [2,3], and at universities in England and Australia. These grading systems provide a method of objectively grading student programs automatically for the instructor. By utilizing the computer as an extension of his teaching methods, an equal amount of attention in evaluating class programming work is possible.

B. OBJECTIVE

The objective of this thesis is to investigate the concept of machine grading systems and to provide a practical implementation of a grader for beginning assembly language students. The fulfillment of this objective is accomplished by an analysis of grading techniques and parameters available for evaluating a program. The second element of this objective is accomplished by establishing an operating system specifically designed for assembly language instruction at the Naval Postgraduate School and constructing a machine grading program which will operate within this system and produce graded student programs. Through the courtesy of Professor Andries van Dam of Brown University, the Brown University Student Operating System was made available to the Naval Postgraduate School for use in the instruction of assembly language.

C. METHODOLOGY AND ORGANIZATION OF THESIS

The methodology of this investigation was to first review previous work in the field of machine grading. The metrics of a program which can be utilized in its evaluation are discussed in Chapter II. A summary of the essential properties of an automatic grading system is presented at the end of Chapter II. The description of a grading system and its implementation is given in Chapter III. In addition, grader use with student programs is discussed. The conclusions reached after the investigation, and the recommendations for further study are contained in Chapter IV.

The first step in constructing a grader system was to revise the structure of the Brown University Student Operating System and implement it at the Naval Postgraduate School. The Student Operating System was then examined for inherent characteristics which provide the instructor with teaching and grading capability. An appraisal of this system with respect to grading capabilities is presented in Appendix A. The other appendices contain various computer programs, flowcharts, and outputs relating to this study.

II. CHARACTERISTICS OF A GRADER

A. REVIEW OF AUTOMATIC GRADING SYSTEMS

A search of the literature for reports of automatic grading techniques produced several documents specifically addressing the subject. Perhaps the first published report of grader programs is that of Hollingsworth [1]. This grader was used at Rensselaer Polytechnic Institute in 1959 and was implemented on an IBM 650 computer. The author states that the university could not have accommodated programming classes of 80 to 120 students without the use of the grader. This early version of an automatic grader provided batch processing and grading of student jobs for a variety of exercises.

Students submitted new programs and corrections to old programs for keypunching at the end of a class period. After keypunching, the corrections were filed with the old programs and all the programs, old and new, were arranged by exercises. The grader program deck was inserted with the student jobs. Considerable effort in hand filing punched cards was necessary in this system.

This grader punched identification cards for student programs, checked results and punched either WRONG ANSWER for an incorrect solution or PROBLEM COMPLETE for all correct answers.

A great deal of operator control was required to process student programs. Manual re-entry for overflow, invalid addresses, and other problems was required. A limitation of this grader was the fact that it processed only machine language programs. Student programs could modify the grader itself, causing additional problems.

The grading system required the specification of four addresses associated with the exercise, viz., the location of the entrance to the student program, the location to which the student program must exit, the address of variables and the address of results.

An advantage of this system was that programming classes were not required to work on a single project at the same time. Since various exercises could be graded during one batch run, students could work on several projects simultaneously if they so desired. Despite some difficulties with this system, Hollingsworth [1] indicated that the grader had more than justified the effort in its implementation.

Automatic machine grading programs in use at Stanford University are reported by Forsythe and Wirth [2,3]. Two ALGOL grading programs have been used in connection with introductory ALGOL programming and numerical analysis courses at Stanford University since 1961. One grader simply furnishes data and checks answers. The other program provides a searching test of the reliability and efficiency of an integration procedure. The major difficulties and limitations of the Rensselaer grader were overcome in the grading systems in use at Stanford. The concept of basing a grade only on the binary answer "CORRECT" or "WRONG" was used for beginning students. More advanced students were graded by the second grader on their ability to solve the integration problem efficiently and with some degree of reliability. The authors claim that this grader marks a further step in the automation of grading because the quality of the program is evaluated.

R. E. Berry [4] examined the use of automatic grading programs for checking the practical work of students in a numerical analysis

course. This report provides a comparison of the Forsythe and Wirth program [2] and a grader program by Naur [5]. Both of these graders considered the same problem (finding the root of a given function to a given accuracy in a prescribed interval).

Naur's method required that the student program be included in the grader program as a labelled block of coding, accessed by means of a switch. The grader assigned values to variables which were global to the block. In this manner different problems could be presented to the student algorithm. When a solution was achieved, a branch to a given label within the grader program was made and an examination of the result was conducted. Further problems were then presented until the problem list was exhausted and the grader continued with the next student program. In contrast, the ALGOL grader implemented at Stanford [2] was called by the student program. The students were required to write a procedure with parameters in a specific order. Each procedure was tested by supplying it with different sets of parameters and checking the solution obtained.

One of the difficulties of grading was well illustrated in the work by Berry [4]. The problem of evaluating the quality of an algorithm for a problem which has various methods of solution is exceedingly difficult. The author states that in trying to present a problem to which different methods of solution are available, the possibility of successfully grading the work presented is remote.

Berry points out that the demand for automatic grading programs in Great Britain is not expected to be substantial, since computer classes are smaller. One commendable aspect of graders was verified by both Naur [5] and Berry [4]. They arouse interest and provide

an incentive to work. Students submitting programs to this type of grader found the challenge of writing the best procedure an incentive in their work.

The first known grading procedure for PL/I exercises was implemented on an IBM System/360 model 50 computer at the Australian National University in 1966 [6]. The decision to use automatic grading by computer of all student attempts at class problems was based on the following:

1. The large volume of work required for teachers to carry out grading and associated record keeping.
2. The apparent difficulty of teaching input/output sufficiently early in the course for students to begin writing programs.
3. The ease with which input test values could be supplied to the students and results printed under control of a grading procedure.

The idea of using a grading system to bypass difficult programming aspects during beginning phases of instruction was not found in earlier documentation of graders. Temperley and Smith [6] also point out that the use of PL/I made possible the evaluation of results in a variety of formats. Test data and results could be in the form of arrays, structures, bit-strings and character-strings as well as scalar numeric values.

The timeliness of this investigation is borne out by a recent paper on an automatic grading scheme for simple programming exercises by Hext and Winings [7]. This report documents the development of the Basser Automatic Grading Scheme (BAGS) at the University of Sydney, Sydney, Australia. A major difference in the BAGS system is that it can handle exercises in several different languages. No special

action by computer operators is required with this scheme. Another significant departure from previously documented graders is the structure of the system itself. Whereas most other schemes embed their exercises in some larger program, BAGS is part of the standard operating system and its exercises are run under batch processing methods.

The basic requirements of the program were as follows:

- " (1) It should handle exercises in ALGOL, in MINIGOL (a subset of ALGOL) and in KDF 9 Assembly Code.
- (2) It should not place any additional burden on the operators.
- (3) It should record every attempt at an exercise, with sufficient data for calculating a mark.
- (4) It should provide summaries on request for specified classes and exercises over a given period." [7]

The method of grading in the BAGS system is unique. The assignment of credit for an exercise not only takes into account the mark for each program attempted but also the number of attempts to solve the problem. The maximum mark for a single exercise was set at five. One point was credited for each of the following:

- a. The program compiled successfully.
- b. The program ran to completion.
- c. The first answer was correct.
- d. The second answer was correct.
- e. The program ran successfully within a prescribed central processor time.

The graded results were printed for each student in the form of i/j. This grade states that the student made i attempts at the exercise and that his best mark was j. If the student's program satisfied all five criteria the first time, a mark of 1/5 was assigned. A grade was assigned by the formula:

$$\text{GRADE} = \text{average of } \frac{100 \times j}{4 + i}$$

The average was taken over all exercises being marked.

The original version of the Brown University Student Operating System included a grading system. The SOS grader has subsequently been eliminated from the SOS system at Brown University. In a search for properties of a grader it is revealing to analyze the characteristics of the original SOS grader to determine the cause of its demise. The operation of the original SOS grading system is described as follows:

"Using appropriate supervisor calls, the student communicates answers and messages to the grader to indicate the problem to be graded (initialization), the answers and error exits to be generated, and to obtain special information dependent on the specific program being graded. In addition to communicating with the grader while his program is being executed, the student must write his program following certain conventions: 1) various sections of his program must bear standard labels as indicated in each problem assignment; 2) the data is read in the usual manner, but must be placed in an assigned location; and 3) the expected answer must be in the location and form as specified in each problem assignment." [10]

As can be seen by this description a great deal is required of the student to ensure that his program is processed properly by the grader. After attempting two different versions of the grader at Brown University, it was found that the student programs were required to initiate almost all communication with the grading program and doing this made the student programs both more difficult to write and more difficult to make efficient, due to the rigidity of the communication conventions. A second, and fatal, limitation of this system was the inability to specify a programming problem explicitly enough to allow the student to follow grader conventions without actually specifying the method of solution. A review of two student programs which had been graded by this system and

discussions with the authors of the system verified these problem areas as the reasons for elimination of the grading feature from the Student Operating System at Brown University.

It is evident that significant improvements in automatic grading techniques have evolved since the development of the Rensselaer grader in 1959. In the previous discussion of work in this field an attempt has been made to chronologically illustrate this evolution. Reference 7 states that 1500 first-year students will be enrolled in programming classes at the University of Sydney in the next two terms. Under these circumstances, the value of grading systems seems well established. However, much of the documentation on this subject has concerned implementation of a particular grading program. The next section proposes certain metrics of an algorithm which might be used in a grading system. The effectuation of some of these ideas may not be practical at present. Nevertheless, with the increased interest in automatic grading an exploration of conceivable characteristics of a grader is deemed necessary.

B. THE METRICS OF AN ALGORITHM

The term Grader Program is perhaps a misnomer; not all such programs assign a mark or grade to the work submitted. A grading system is sufficient if the program can provide enough information for a mark to be quickly assigned by the student's supervisor. The basis for the output of this information is discussed in the subsequent paragraphs.

It is advisable to discuss the upper and lower bounds of a grading system's performance. For instance, it is a well known fact that there is no algorithm to determine whether or not a given procedure

is total. That is, a grader program cannot determine whether another program contains no loops. An upper limit in the performance of a grader program might be the class of functions which fall in this category. A second class of characteristics of a grader which may limit achievable performance, is the class of functions which are computable but are impractical to implement. Berry [4] has shown that considerable complication results in grading a problem in which any one of several methods of solution might be acceptable. A lower bound in the performance of a grader is simply that it must produce some information about the algorithm being tested. A program which recorded the trivial fact that a student program had been processed provides information which might be used to evaluate a student's progress even though the program itself is not evaluated. Therefore, it can be concluded that there exists a range in which a grader must operate to produce the basis for the evaluation of an algorithm.

The most common measure used in the evaluation of a student exercise is the determination of whether or not the solution is correct. This measure may provide a sufficient basis for a grade for introductory programming exercises [1]. However, it is suggested that this is dependent on the characteristics of the problem assigned. A grader should be capable of providing information pertaining to the accuracy of a program result. In addition, a grading program should be capable of measuring this value in a variety of formats such as in the BAGS grading system [7].

Criteria for evaluating a program which have been used are successful assembly, successful execution, and the number of

attempts at an exercise [7]. Such criteria form an absolute judgement of a program in that the evaluating process does not consider the quality of the program.

A grader which only evaluates the results of another program does not provide specific information about the program itself. Qualities of a program which have significance with respect to grading are execution time, storage utilization, programming techniques, program logic and annotation. These characteristics form a basis for the measure of the quality of a program.

The completion of a programming exercise within a sufficiently short central processor time was one criteria in the BAGS method of grading [7]. This measure of a program can provide the instructor with a means of evaluating program efficiency. Application of this measurement is more appropriate for advanced programming classes.

The amount of storage utilized by a program is a characteristic which can be used to evaluate its quality. This measure is all too often neglected in programming classes but has direct application in the field. Incorporation of this measurement in a grading system can be an effective means of stressing the importance of the efficient use of storage.

If an instructor compared a program which contained one hundred ADD instructions to sum one hundred numbers with a program which used a single indexed ADD instruction in a program loop, the latter would undoubtedly be considered superior. Although programming techniques vary considerably, the quality measure of a program may provide the best criteria for evaluation of a programmer's efforts. Due to the fact that good programming techniques decrease execution time and

provide efficient use of storage, this measure is more effective than analysis of time or storage parameters alone. The multiplicity of thought forms a substantial barrier to the implementation of this characteristic in a grading system. A specific problem could possibly be designed which then would be inspected for key variables in a prescribed optimum order by a grading system. It appears that the implementation of such a system would require the development of an extremely capable heuristic pattern matching algorithm. It is doubtful whether the effort would be justified in obtaining this measurement.

A definition of program logic is given by Montalbano [8]. The subject matter of program logic is the matching of appropriate actions to given conditions. For example, program logic is concerned with such statements as "if a given record is an inventory adjustment, process the record before all other records affecting this stock item; if the record is a receipt, process it before any sales order affecting this stock item." In essence, program logic is the effective description of computer outputs as functions of inputs. Techniques such as narrative description, program comments, flow charts, decision tables and system specification languages of the Iverson type are tools for expressing program logic [8].

The logical structure of an algorithm is another measure which forms a basis for the evaluation of the program. The evaluation of the logic of the program is extremely difficult because of the variety of paths which a programmer may take to solve the problem. Chapter IV describes the implementation of a grader which can evaluate subsections of a program independently. The logical

order in which a solution was attempted might be measured in the broadest sense by verifying the result of each sequential section of a program. A method to compare program logic descriptions such as those described in [8] would be an invaluable grading technique. At the present time an inspection by the instructor of program logic descriptions such as flow charts is required.

Higher level languages have an advantage over assembler languages in that they possess a "self-documenting" feature [9]. The documentation of assembly language programs is an important area which requires evaluation. The annotation of a program varies from individual to individual. The basic rules of documentation for assembly language programs are described by Opler [9]. It is doubtful if a machine process could automatically produce an evaluation of a student program with respect to these rules. This author believes that graphic methods may prove to be an effective way of quickly evaluating the logic and the documentation of a program. A correct logic flow chart might be superimposed on a student flow chart on a display device. Key decision blocks could then be checked by the instructor. Spot checking of annotation can be accomplished on a graphic display device if the student programs are disk resident such as in the SOS system (Appendix A).

The concept of providing automatic methods which will evaluate the quality of a program is in its infancy. It is expected that further work in this field will incorporate some of the previously listed metrics of a program.

C. GRADER DESIGN CONSIDERATIONS

The first consideration in designing a grading system should be the specification of objectives. What type of information is necessary to evaluate the programs in a particular course? The designer must concern himself with the previously mentioned parameters of a program which are required to be measured.

The second step in the design of a grader is to carefully investigate the system available for grader use. Forsythe [3] reports that the Stanford grader was possible only because of the well-designed BALGOL compiler with its own compiler-with-library generator, and because a procedure called BUTTERFLY could generate relocatable machine-language programs. It was found that neither the IBM 7090 BALGOL system nor the Burroughs B5000 ALGOL 60 system were well adapted to the grading problem at Stanford University.

One of the most critical characteristics in the design of a grading system is the amount of involvement of the student with the grader itself. The student should not be required to alter his program significantly to conform to grader conventions. A minimum of communications between the grader and the student, with maximum communication capability between the instructor and student should be the rule in a grading operation.

Another feature of a grading system which must be considered is the work required by the instructor to program the grader to process the specific problem. The grading system should include macro-instructions for instructor messages and general communication sections for the passing of answers and data. In general, instructor involvement with the grader should be held to a minimum.

The student should be able to work out solutions using his own data, but data should be provided by the instructor in some manner at grading time. Grading of the program with data from the instructor could present situations which had not occurred to the student.

A grading system should be able to handle a variety of problems. Programming a complete grading system to evaluate a specific problem would be inefficient. Manual intervention by computer operators to process grading jobs is impractical on present large scale computing systems such as the IBM/360. Execution time of the student program should not be significantly increased when executed for grading. Additional characteristics such as the ability to evaluate programs written in different languages, and the capability to change problem parameters with minimum effort should be considered in the design of a grading system.

A summary of proposed grader design characteristics is presented in Table I. This list is a summary of features which might be considered in the design of a grading system.

These characteristics of a grading system have been proposed as the result of study of documented grading systems and also from limited experience gained in implementing a practical grader. Such features, if only achieved to some minimum degree, will provide a grading system which can assist the instructor greatly in programming instruction classes. The next chapter discussed the mechanization of a grading system which possesses most of these characteristics and is primarily designed for use in beginning assembly language programming courses.

TABLE I

SUMMARY OF GRADER DESIGN CHARACTERISTICS

A. EVALUATION CAPABILITY

<u>CHARACTERISTIC EVALUATED</u>	<u>TYPE OF MEASURE</u>
Result of Algorithm	Absolute
Tolerance of Solution	Absolute
Successful Assembly	Absolute
Successful Execution	Absolute
Number of Attempts at Exercise	Absolute
Assembly Time	Quality
Execution Time	Quality
Number of Instructions Executed	Quality
Storage Utilization	Quality
Program Logic	Quality
Method of Solution	Quality
Documentation	Quality

B. MINIMUM STUDENT COMMUNICATIONS WITH GRADER

C. MINIMUM INSTRUCTOR EFFORT IN USING GRADER

D. CAPABILITY TO GRADE DIFFERENT LANGUAGES

E. CAPABILITY TO GRADE A VARIETY OF PROBLEMS

F. AMOUNT OF OPERATOR ATTENTION REQUIRED

G. EFFORT REQUIRED TO ALTER OR ADD PROBLEMS TO GRADING SYSTEM

H. DATA MANIPULATION FOR PROBLEM DURING GRADING RUN

III. IMPLEMENTATION OF A GRADER

A. PURPOSE OF GRADER

Since the Student Operating System (Appendix A) is well-suited for beginning assembly language programmers, a grading program for use in introductory programming classes was written. The primary purpose of this program is to check student answers and assign a grade. It is designed to be used for grading small projects assigned to students who are just learning the language. With a grading system to handle numerous small projects, concentration on one particular area such as indexing, looping or arithmetic operations is possible and a project can be assigned to cover that subject. This type of grader allows the instructor to assign more projects and gives the student an opportunity to obtain machine experience on a wider variety of assembly language programming aspects. This would not be permissible with a system which was designed for a specific problem of a more complex nature.

The grader is an absolute type of system in that it operates on answers from the student. In addition to checking answers, communications from the instructor to the student have been included as an essential part in the design of this grader. In summary, the purpose of the grader is to allow the instructor to assign numerous small projects in an introductory assembly language course, to grade the projects and provide messages from the instructor to the student. By using the computer as a tool for evaluation of class work, the instructor can provide an equal amount of attention to appraising each student's progress.

B. MECHANIZATION OF THE GRADER

Although the Brown University Student Operating System Grader program has been eliminated from the system, the structure for including a grader was intact in the system received. By choosing the GRADER=YES option at system generation time, communication links, control blocks, and initialization of parameters for a grader were generated within the SOS system. Through the use of parts of this available structure and modifications to o. elimination of some of its features, it was possible to concentrate efforts on the grading program itself.

The grader operates under control of the SOS control program, and is called by the student when he feels that his program is correct. The basic structure of the grading program and its interrelation with the rest of the SOS system is shown in Chart Number 1, Appendix C. During initialization of a student's program the SOS control program calls the bookkeeper. The bookkeeper sets system options specified by the user. If the student has specified GRADER=YES on his job card, grader parameters are initialized by the bookkeeper and the grader program is called.

A heading indicating that this is a grader run is printed and the SOS interpreter is then called from the grader. The SOS interpreter executes SOS machine code interpretively. The first two instructions of the student's program are executed setting up initialization of the grader for the particular problem specified by the student.

After grader initialization, the interpreter continues execution of the student's program until it is desired to have an answer checked. The student issues a supervisor call and indicates

the location of his answer. The main section of the grader checks the answer and assigns a grade, substitutes the correct answer if the instructor so desires, prints the results and any messages from the instructor if the answer was not correct. Control is passed to the interpreter again for further execution of the student's program. This technique of grading the student's program during execution continues until the final answer is graded. Return to the control program is then made and the next student job is processed.

The grading program has an initialization section to initialize the grader for a specific problem. The main section of the grader handles all aspects of grading the problem. A communications section for passing answers and messages from the instructor is included. The grader is designed to grade up to five intermediate answers for a problem and a final answer. The capability of grading twelve different problems is included in the grading program. Six different answers should suffice as a maximum for beginning programming projects. The capability to grade twelve different problems allows the instructor to assign one problem per week during the class quarter.

The first part of the grading program establishes addressability with the other major sections of the SOS system and saves the return address to allow return to the control program at the end of the grading run. A grader heading is printed and the interpreter is given control. Execution of the first instruction in the student's program starts the initialization of the grader. The return address to the interpreter is first saved and the program control counter is incremented to point to the next sequential instruction in the student program. A check for student core wrap around is made and a

check for a proper initialization call is accomplished. A halfword grader control block is used throughout the program to set various conditions. The first byte of this control block is used by the system. The bookkeeper sets a bit on if the student has asked for a grading run. The grader initialization section turns a second switch on if grading is in process. The second byte of the grader control block is used by the instructor to control various functions of the main grading section and its use will be described in the section on use of the grader.

If a grading run has been requested and initialization has not yet taken place, the data for the problem specified is placed in a general communication section. This is accomplished by the use of a dummy section which describes the layout of storage for the data that the instructor has defined for a specific problem. Symbolic names defined in the dummy section are then used as operands in the main grading section during the grading process. This method allows the instructor to separately assemble the appropriate data in a control section and use the link editor to link his answer CSECT in object form to the system.

Various error routines and messages are included in the grader initialization section. Upon completion of initialization, a message to that effect is printed and execution of the student's program is continued. The grading program grades the result of up to five intermediate sections which the student has used in his program to achieve a final answer to a specific problem. When the student has reached a particular result, the main grader section of the grading program is called. This section first determines which

answer has been specified, and then branches to the appropriate routine. The grading routines for intermediate answers are similar except for correct answer usage data and the messages sent to the student.

The intermediate answer grading routines first obtain the student's answer. This is accomplished by the subroutine called GETANS. The address of the answer in student core, which is word-oriented, is obtained and converted to a System/360 address. The correct answer specified by the instructor is then compared with the student's answer. If the answer is correct, a message indicating this fact is printed and credit for that answer is accumulated. The accumulated credit for each intermediate answer is also printed. If the student answer is not correct, a test is made to see if the instructor has a message for the student which might assist him in obtaining the correct answer. If the instructor has included a message, it is printed on the student's output. The instructor has the option of substituting correct data for a wrong answer if he so desires. Thus the subsections of a student program can be checked even if the program as a whole computes incorrectly. If the instructor has chosen this option, the correct data is moved into the student core address of his incorrect answer. No credit is accumulated for a wrong answer. A message indicating that the answer was not correct is printed.

If there is just one answer for a particular problem or if a final solution has been reached, the final answer routine of the main grading section is entered. The final answer routine checks the solution and prints a message indicating whether it is correct or not. A total grade is then determined. Fifty points are given

for a correct final answer. Each of the five intermediate answers are worth ten points. A grade for a problem with five intermediate answers and a final answer is totaled by summing the accumulated intermediate answer credit and the credit for the final answer. If the problem required less than five intermediate answers, the accumulated credit for the intermediate answers is first supplemented to fifty and then added to the credit for the final answer. The assignment of these credit values for intermediate and final answers is arbitrary and may be changed by an instructor with only minor modifications to the grading program. The total machine grade and any final message to the student is then printed on the student's output.

The next part of the grading program evaluates the efficiency of the student's program to some extent. The number of instructions executed in the student program is printed. The mean number of instructions executed in previous programs for this project is supplied by the instructor in his answer control section. These two values are compared and additional credit is given to the student whose program required the execution of fewer instructions. A message indicating whether the student required more or less than the median is printed. The total machine grade with bonus credit is then printed. This capability is optional and is set by the instructor in the answer control section.

A return to the SOS control program is then made for further processing of student jobs. The grader program is included in Appendix B. Flowchart documentation of the program is included in Appendix C.

C. USE OF THE GRADER

This section describes the conventions which must be followed by the student and the instructor to ensure proper operation of the grading system. Examples of graded student programs are shown in Appendix D. The control section established by the instructor to communicate answers and messages to the grader is also included as Appendix E.

1. Student Grading Conventions

The student is required to indicate four items to the grader. He must first specify that he desires his program to be graded. The student then indicates which problem is to be graded. When the student has reached a solution to the problem he must indicate to the grader which of the six possible answers he wants to have checked and the location of that answer.

a. Calling for a Grading Run

When the student feels that his program is ready for grading, he utilizes the JCL bookkeeping parameter option feature of the SOS system to call the grader. This is done by placing GRADER=YES on the job card starting in column 26.

EXAMPLE:

Column:	1	6	16	26
	/JOB	BACONRF	SOSJOB1	GRADER=YES

b. Specification of the Problem to be Graded

To indicate which problem is to be graded the student places as the first executable instruction of his program the supervisor call SVC 3,0 and a second instruction DC (decimal number 0 through 11) immediately following the supervisor call. The supervisor call initializes the grader for the problem specified in the

second instruction. For example, if the student desired problem number three to be graded, the first two instructions of his program would be as follows:

Column:	10	16
	SVC	3,0
	DC	2

c. Specification of the Answer to be Graded

The student must indicate to the grader which answer is to be graded. This is accomplished by issuing a supervisor call.

EXAMPLE:

<u>SUPERVISOR CALL</u>	<u>SPECIFIES THIS ANSWER</u>
SVC 3,1	First intermediate answer
SVC 3,2	Second intermediate answer
SVC 3,3	Third intermediate answer
SVC 3,4	Fourth intermediate answer
SVC 3,5	Fifth intermediate answer
SVC 3,6	Final Answer

d. Specifying the Location of the Answer

Immediately following the supervisor call specifying the answer, the student places a HALT instruction with a single operand which is the symbolic name of the location of the answer.

An example of a student requesting a check of his fourth intermediate answer which had been previously stored in a storage location called ANSWER would be as follows:

Column:	10	16
	SVC H	3,4 ANSWER

2. Instructor Grading Conventions

The instructor must indicate four items to the grading system: the number of answers to be graded; the correct answers; messages to the student; and grader options.

a. Specification of Answers

To pass correct answers to the grader, the instructor defines fullword constants through the use of a macro-instruction called ANS. The answers will be defined in the instructor's control section.

EXAMPLE:

Column:	10	16
	ANS	8,13,-99,33397

The above example defines three intermediate answers and a final answer. Answers must be specified in order.

b. Specification of the Number of Answers

The number of answers to be graded is defined as a fullword constant in the control section.

EXAMPLE:

Column:	1	10	16
	NOINTANS	DC	F'decimal number 0-5'

- c. Specification of the number of instructions executed.

The instructor specifies a value for the grader to use in the comparison of instructions executed by the student's program by defining a fullword constant in the control section.

EXAMPLE:

Column:	1	10	16
	NOX	DC	F'decimal number'

- d. Specification of grader options

The grading system has three optional features. Messages to the student, the substitution of correct answers, and the assignment of additional credit are chosen by the instructor through the use of a macro-instruction called PROF. If it is desired to include all three features the default is chosen as follows:

Column:	10	16
	PROF	(blank operand field)

Operands to eliminate these features are:

NOMSG - eliminates message feature
NODATA - correct answers are not substituted
NOBONUS- feature to check the number of instructions
executed and assign bonus credit is removed

EXAMPLE: To prevent the substitution of correct answers and eliminate the assignment of additional credit the instructor writes:

Column:	10	16
	PROF	NODATA, NOBONUS

Note: The operands may be placed in any order in the operand field.

e. Specification of messages

The instructor may send a message to the student whenever an incorrect answer is obtained and at the end of a grading run. These grader messages are defined through the use of a macro-instruction called MSG. The instructor may define as many as six messages in the control section. The labels for the messages are IMESS1 through IMESS5 for intermediate answer messages and IMESSF for a final message. Messages can be up to 132 characters in length.

EXAMPLE:

Column:	1	10	16
	IMESS4	MSG	'message to student'

Note: This message would be printed if the student's fourth intermediate answer were incorrect.

The order in which these items must be defined in the control section is shown in the example in Appendix E. The instructor specifies these parameters in a control section with labels which indicate the particular problem. GINITO indicates the first problem

whereas GINIT11 indicates the twelfth problem. The control section can be assembled and an object deck obtained using the System/360 ASMA procedure. The control section may then be linked to the SOS system by the linkage editor.

D. FEATURES AND LIMITATIONS OF THE GRADER

A comparison of the characteristics of this grader with the general properties of grading systems listed in the previous chapter is now presented. The SOS grader accomplishes the basic objective of determining whether an answer is correct or not and assigning a grade.

The student involvement with the grading system itself has been kept to a minimum. Very little is required of the student to indicate the problem to be graded and correct answers.

The student can be assisted in his programming by messages from the instructor. This feature allows up to six messages from the instructor to the student. These messages are predefined to fit the circumstances of the problem solution.

Through the use of a general communications section, ease of specifying the answers and messages for various problems is made possible. Macro-instructions for use with the answer control section allow the instructor to provide grader data with minimum effort.

A feature of the grader is its flexibility in grading as many as six answers for one problem and its ability to grade twelve different problems during one batch run of student jobs. Students may work on several projects at the same time because of this capability.

The substitution of correct intermediate answers allows independent grading of the subsections of the student's program. By evaluating each section of a program separately, a more representative grade can be assigned.

The only feature of the grader which examines the quality of a student's program is the examination of the number of instructions executed. A better measurement of the efficiency of the program would be the execution time of the student program. The grader does not evaluate storage utilization or the number of runs submitted prior to grading. Inherent characteristics of the SOS system can be used by the instructor to assign a grade based on these parameters. These characteristics are discussed in Appendix A.

The SOS system uses word-oriented storage. Because of this, the grader is designed to evaluate fullword answers. A minor limitation is the fixed credit structure of the grader. However, this can be modified to suit a particular instructor's desires. Since the grader is designed for use in beginning programming classes, these limitations do not affect the capability to grade the type of program expected in introductory class work.

IV. CONCLUSIONS AND RECOMMENDATIONS

A. CONCLUSIONS

A grading system for use with the Student Operating System in beginning assembly language classes has been implemented at the U.S. Naval Postgraduate School. This system will significantly aid the instructor who must grade student exercises in classes with a large number of students.

This type of grader will provide the capability to assign more exercises during the academic quarter. Programming instruction can emphasize various aspects of the language in each assignment, thus allowing the student to gain more experience in assembly language programming.

Since the evaluation of a programming exercise is inevitably repetitive, the concept of writing a grader program which would be able to check other programs is both practical and useful. Documentation of previous grading systems has emphasized the implementation of the grading program. This investigation has examined the overall concept of a grading system. Characteristics of a grader have been evaluated. Several of these characteristics which measure the quality of a program would be extremely difficult to implement at the present time. Despite these complexities, it is anticipated that grading systems of the future will concentrate on the evaluation of the quality of a program.

A significant quality of grading systems in general is the fact that the machine may be more objective in grading than the human, because of its notable lack of prejudice and its inability to become bored [3].

Grading systems generate a paradoxical situation. A system which is designed to minimize student involvement with the grader itself, invariably requires more effort on the part of the instructor. A grader which not only checks answers but also evaluates the quality of the program may require considerable additional labor by the instructor. Because of this situation, tradeoff between student requirements and instructor efforts must be fully evaluated prior to the design of a grading system.

The grading system should allow the instructor flexibility in generating class assignments by minimizing programming work on his part with respect to the grader. It should be recognized however, that once the initial effort is expended in generating the problem, it can be utilized many times.

When a grading system is to be programmed, it behooves its designer to thoroughly examine the system with which it will interact. The communication capabilities and limitations of the system must be evaluated. The system parameters such as execution time, the number of instructions executed, and storage limitations should be examined before the initial design of a grading system. In this manner, various existing facilities may be used in the grading system.

B. RECOMMENDATIONS

It is strongly recommended that the grading system be used at the Naval Postgraduate School. It represents a powerful aid to assembly language programming instruction.

Recommendations for augmentation of the grader are:

- (a) Inclusion of the standard data option of the Student Operating System to be used in conjunction with the grader.

- (b) Revision of the Student Operating System statistics keeper so that it can produce a grade book based on results obtained from the grader.
- (c) Modification of the grader by replacing student supervisor calls to the grader with simple Student Operating System macro-instructions.
- (d) Provide the Student Operating System with a meaningful timer and revise the grader to utilize execution time as a measure of the efficiency of student programs.
- (e) Assignment of these recommendations to students as suitable term projects in an assembly language programming class.

Very little work has been done in developing effective techniques to automatically evaluate the quality of a program. It is recommended that further investigation of both the theoretical and the practical phases of this area of grading be conducted.

APPENDIX A

The Student Operating System

A. PROGRAMMING INSTRUCTION AT THE POSTGRADUATE SCHOOL

Students taking a beginning Computer Science course at the Naval Postgraduate School have ample opportunity for actual machine experience. This is not the case for some universities where cost considerations take precedence and research work usually justifies the existence of the university computer center.

To add credence to the idea of efficiently instructing programming courses, an indirect measure of the amount of programming instruction at the Postgraduate School was obtained. This measure is the number of WATFOR jobs processed. WATFOR is an acronym for the Waterloo FORTRAN compiler. This system has been used extensively for introductory FORTRAN programming classes since October, 1967.

Table II presents the Naval Postgraduate School Computer Facility usage data. Approximately forty percent of the total number of computing jobs submitted in the past eighteen months have been WATFOR jobs. This is an indication that a primary use of the Computer Facility is programming instruction. Many of the jobs listed in the STUDENT category of Table II are also programming instruction class work. Faculty programming jobs comprise 8.75 percent of the total and are assumed to be primarily associated with research studies. This evidence indirectly substantiates the fact that a major use of the Computer Facility is programming instruction.

TABLE II
U. S. NAVAL POSTGRADUATE SCHOOL COMPUTER FACILITY USAGE DATA
(NUMBER OF COMPUTING JOBS SUBMITTED)

MONTH/ YEAR	STUDENT	FACULTY	SOURCE OF INPUT			TOTAL
			COMPUTER FACILITY	MISCELLANEOUS	WATFOR	
MAR 69	8,395	2,793	351	300	5,083	16,922
FEB 69	11,455	1,121	534	260	4,142	13,370
JAN 69	5,340	960	550	216	2,968	7,066
DEC 68	3,763	461	181	60	1,977	4,465
NOV 68	14,808	775	371	62	7,940	16,016
OCT 68	12,032	969	442	167	7,227	13,610
SEP 68	10,427	673	461	178	2,866	11,739
AUG 68	14,420	838	538	206	8,031	16,002
JUL 68	12,179	796	512	194	8,242	13,681
JUN 68	4,978	826	502	174	2,210	6,480
MAY 68	17,475	771	566	193	8,702	19,005
APR 68	13,384	736	493	96	6,111	14,709
MAR 68	9,777	853	698	116	4,848	11,444
FEB 68	8,239	1,175	496	54	4,013	9,964
JAN 68	4,519	1,066	483	65	1,718	6,133
DEC 67	4,068	740	351	61	1,489	5,220
NOV 67	9,582	1,306	398	56	2,577	11,342
OCT 67	7,366	1,183	348	77	2,065	8,974

B. THE ORIGINAL STUDENT OPERATING SYSTEM

A system which can be used for assembly language instruction is the Brown University Student Operating System [10]. This student operating system provides an assembler which produces code for a simplified machine, an interpreter which simulates the simplified machine and a control program for storage manipulation, editing of the student's program and statistics gathering. Wile [10] documents the capability of this system to provide assembly language teaching and processing at costs comparable to other university algebraic compilers used for student programming instruction.

The Student Operating System (SOS) offers several advantages to the instructor who wishes to utilize the IBM System/360 in an assembly language class. The feature of changing basic parameter settings at system generation time or taking default options allows the instructor to change the capabilities of the entire system for particular programming projects. All of the system is in core at all times and set up for each student job requires less than twenty machine instructions. SOS utilizes a simplified job control language, thus practically eliminating the difficult task of explaining the use of and reason for the complex, but extremely powerful, general purpose JCL required by the System/360 Operating System. Student jobs are cataloged on a disk; editing is done directly on the cataloged programs, thus introducing the student to the concept of text editing and library maintenance as well as reducing the volume of cards which the computer is required to read.

The SOS machine and assembly languages were designed to eliminate some of the more difficult programming concepts inherent in the

System/360 machine and assembly languages [10]. However, the SOS processing languages provide a compatible introduction to the related System/360 languages as the SOS assembly language mnemonics are, in general, a subset of the System/360 mnemonics. Arithmetic and logical operations are performed in a similar manner to System/360 operations. Data is in two's complement form and operations are performed in registers producing results and interruptions similar to 360 instructions.

The instructor may view the elimination of base/displacement addressability considerations, variable length instruction and data formats, floating point, packed decimal and character format conversions, relocatability, control and dummy section capabilities, and condition code testing from the assembly language repertoire as being too restrictive for the proper introduction of assembly language programming to the student. However, it is the opinion of this author that provisions included in the SOS system for indexing, indirect addressing, register addressing, overflow testing, and all arithmetic and logical operations are adequate to accomplish the objective of assembly language instruction at this school. A reasonable statement of this objective is to provide the student with an understanding of the structure and organization of a computing machine. It is not the intent of this study to document in detail the Student Operating System and its language. Specifications for the system and a description of its use may be found in Ref. 11.

C. IMPLEMENTATION OF THE SOS SYSTEM

The SOS System was obtained from Brown University in October, 1968. Documentation of the system received was negligible. System

generation was accomplished by issuing a single macro instruction (SOS); all program segments were included as lower level macro calls. This version was unsuitable for practical use in implementing a grader program within the system or for making changes to the system in general. The "SOS" macro included system default parameters. Required global set symbols were included in this macro. The main section of the "SOS" macro was the SOS processor common area. This section called the eight major macros in the system. The system assembled as the single macro-instruction "SOS" with this structure.

The IBM System/360 assembly language includes the COPY instruction. This instruction obtains source language coding from a library and includes it in the program currently being assembled. The assembler inserts the requested coding immediately after the COPY instruction is encountered. This instruction was utilized to provide a system structure which allowed manipulation of each of the eight major sections separately. The macro structure of each of the previously mentioned macros was removed and all global set symbols were placed in a section named SOSGBL which was then used as copy text in the SOS macro. Keyword parameters in the SOS macro were modified to make them compatible with the revised structure. In addition, the SOS macro was shortened to include only the necessary set symbol instructions. The original SOS processor common area was modified to call in the eight major program sections of the system by the use of COPY statements instead of the previous macro-expansion method. This section of coding was renamed SOSMAIN. The modified version of the system was generated on 7 February 1969.

D. SOS CHARACTERISTICS FOR STUDENT EVALUATION

Several inherent characteristics of the SOS system interrelate with the concept of grading a student's programming work. In describing the operation of a grader in use at Stanford University, Forsythe and Wirth [2] state that it is inappropriate to grade beginning students on the execution time of their programs or to evaluate the amount of storage used by their programs. Certainly the novice programmer should not be severely restricted in his programming efforts by these parameters. However, he should be made aware of their existence and thus eliminate the breeding of poor programming habits at the outset of instruction. By using the capability of varying system parameters in the SOS system the instructor may impose restrictions which will require the student to consider such aspects of programming as storage utilization, the number of instructions executed, register usage, and the number of program runs to achieve a solution.

The capability of imposing restraints on the student is essentially a method of evaluating his ability to program within the restricted environment. It is not an absolute grading mechanism, but rather a circuitous method of examining his programming ability. The features of the SOS system which provide this capability are discussed in the subsequent paragraphs of this section.

The highest configuration of SOS core the instructor wishes his students to use can be set by the keyword parameter MAXCNF = (1/2/..../8) at system generation time or by the bookkeeping parameter CONF = (1/2/..../8) at execution time. The student core size is then set to the value specified times 512 words. An instructor could generate a problem whose solution required approximately 1000 words

of storage and then specify in the assignment that the bookkeeping parameter $CONF = 2$ must be used on the student's job control card. If the student solved the problem within a 1024 storage environment, he could be given additional credit. The poorer student could still achieve the solution but would use the default parameter of 8 (4096 words). The actual configuration used is printed on the first page of the student's output by the bookkeeper. The ability to vary student core size is beneficial to the student because he can be made aware of real-world machine limitations at the start of his programming experience.

One means of student evaluation is to limit the number of programming runs that a student may submit prior to final grading of a project. This method insures that the student will carefully think out each step of his program and thoroughly review each instruction before submitting it for execution.

The SOS system provides the capability to record the number of runs a student submits for execution. If the system is used properly, students will be required to catalogue their programs in the student job library on a disk. All changes to the program are then made through the use of the SOS editing feature. On the first page of the student's output under the data card listing a message is printed "NEXT RUN IS NO.----." This provides the instructor with a means of evaluating the ability of the student to achieve a solution to the problem in as few runs as possible. Again this characteristic of the system does not pass an absolute judgement on a student's solution to a programming project. It does however, provide a better means of evaluating the overall potential of the student.

A third built-in grading characteristic of the SOS system is the capability to vary the maximum number of allowable executable instructions. The primary purpose of this feature is to prevent infinite looping. The maximum number of allowed executable instructions may be set at system generation time by the keyword parameter SOSMAX = (, , maximum number of instructions) or at execution time by the student through the use of the bookkeeping parameter INSC = (decimal number) on the student's job card. This parameter is evaluated directly by the Grader and is discussed in Chapter III.

In summary, the major capabilities of the SOS system with respect to student program evaluation are storage utilization flexibility, the capability to record every attempt at an exercise, and variability of the maximum allowable number of executable instructions. Imaginative and prudent application of these features of the SOS system by an instructor should motivate the student toward good programming practices and at the same time allow a broader evaluation of the true capability of the student.

APPENDIX B

The Grader Program

The Grader Program is shown in assembled form. The Grader Program is assembled with the Student Operating System when the keyword parameter GRADER=YES is chosen at system generation time.

LCC	PROJECT CODE	SCHOOL	STUDENT OPERATING SYSTEM	*** GRADED ***
016570	5230 F120	ADDR1 ADDR2	STMT	SOURCE STATEMENT
016580	07F3	16500	3192	L 3,=V(INTERPT)
			3193	RR 3
016594			9195	SAVF14
016598			9196	PS
016599	00000000		3187	LTORG
			9189	=V(INTERPT)
				ORCP 15

GET ADDRESS OF INTERPT
GO TO INTERPRETER FOR EXEC. OF 1ST INST

LOC	OBJECT CODE	ADDR1	ADDR2	STMT	SOURCE STATEMENT	STUDENT OPERATING SYSTEM	*** GRADER ***
014F2F	59F0 C240		170CC	8248+	LA 15,=A(SOSPRINT)		
014F32	C5EF			8249+	GPRT 4,01,02, .		
014F34	411C C244			8251+	LA 1,=CL12, .		
014F3P	4201 1000		170DC	8252+	MVC 0(2,1),=X,0102,		
016F3F	59F0 C240		171C2	8253+	LA 15,=A(SOSPRINT)		
016F42	C5EF		170CC	8254+	LA 14,15		
016F44	59F0 C234			8255+	ISVCCUT L 14,SAVF=214		
016F4R	C7EF		170CC	8256	RP 14		
					LCAD RETURN		
					CONTINUE STUDENT PROGRAM		
016F44	411C C11A		16FA6	8264	GERR1		
016F4E	59F0 C240		170CC	8265	LA 15,=A(SOSPRINT)		
016F52	C5EF			8266	LA 14,15		
016F54	411C C278		17104	8267	GPRT 19,0F,02, **ABNORMAL E0J**		
016F58	59F0 C240		17116	8268	GPRT 1,=CL12, **ABNORMAL E0J**		
016F62	C5EF		170CC	8269+	MVC 0(2,1),=X,0F02,		
016F64	4004		CC004	8270+	LA 15,=A(SOSPRINT)		
016F68	59F0 C248			8271+	LA 14,15		
016F6C	59EF 0000		CC000	8272	GPRT 14,SAVF=14		
016F70	07EF			8273	LA 14,0(14)		
016F72	4320 C28C		17118	8274	CH 2,=H*6,		
016F76	4720 C6F2		16F7E	8275	CHKSV C		
016F7A	4720 C088		16F44	8276	CH 14		
016F7F	411C C303		1718F	8277	GPRT 17,0E,02, **SVC UNKNOWN**		
016F82	4201 1000		1711A	8278	LA 1,=CL17, **SVC UNKNOWN**		
016F88	59F0 C240		170CC	8279	MVC 0(2,1),=X,CE02,		
016F8C	C5EF			8280+	LA 15,=A(SOSPRINT)		
016F8E	4720 C0C8		16F54	8281	LA 14,15		
				8282	GPRT 82,4F,02, **INITIALIZED ROUTINE SPECIFIED IS NOT AVAILX		
				8283	ABLE, CORRECT YOUR PROBLEM NUMBER. **		
				8284	LA 1,=CL82, **INITIALIZED ROUTINE SPECIFIED IS NOT AVAILABX		
				8285	LA 0(2,1),=X,4F02,		
				8286	LA 15,=A(SOSPRINT)		
				8287	LA 14,15		
				8288	GPRT 82,4F,02, **STUDENT HAS NOT INITIALIZED. CURRENT SVC NOT ACCEX		
				8289	ABLE, CORRECT YOUR PROBLEM NUMBER. **		
				8290	LA 1,=CL82, **INITIALIZED ROUTINE SPECIFIED IS NOT AVAILABX		
				8291	LA 0(2,1),=X,4F02,		
				8292	LA 15,=A(SOSPRINT)		
				8293	LA 14,15		
				8294	GPRT 82,4F,02, **STUDENT HAS NOT INITIALIZED. CURRENT SVC NOT ACCEX		
				8295	ABLE, CORRECT YOUR PROBLEM NUMBER. **		
				8296	LA 1,=CL82, **INITIALIZED ROUTINE SPECIFIED IS NOT AVAILABX		
				8297	LA 0(2,1),=X,4F02,		
				8298	LA 15,=A(SOSPRINT)		
				8299	LA 14,15		
				8300	GPRT 82,4F,02, **STUDENT HAS NOT INITIALIZED. CURRENT SVC NOT ACCEX		
				8301	ABLE, CORRECT YOUR PROBLEM NUMBER. **		
				8302	LA 1,=CL82, **INITIALIZED ROUTINE SPECIFIED IS NOT AVAILABX		
				8303	LA 0(2,1),=X,4F02,		
				8304	LA 15,=A(SOSPRINT)		
				8305	LA 14,15		
				8306	GPRT 82,4F,02, **STUDENT HAS NOT INITIALIZED. CURRENT SVC NOT ACCEX		
				8307	ABLE, CORRECT YOUR PROBLEM NUMBER. **		
				8308	LA 1,=CL82, **INITIALIZED ROUTINE SPECIFIED IS NOT AVAILABX		
				8309	LA 0(2,1),=X,4F02,		
				8310	LA 15,=A(SOSPRINT)		
				8311	LA 14,15		
				8312	GPRT 82,4F,02, **STUDENT HAS NOT INITIALIZED. CURRENT SVC NOT ACCEX		
				8313	ABLE, CORRECT YOUR PROBLEM NUMBER. **		
				8314	LA 1,=CL82, **INITIALIZED ROUTINE SPECIFIED IS NOT AVAILABX		
				8315	LA 0(2,1),=X,4F02,		
				8316	LA 15,=A(SOSPRINT)		
				8317	LA 14,15		
				8318	GPRT 82,4F,02, **STUDENT HAS NOT INITIALIZED. CURRENT SVC NOT ACCEX		
				8319	ABLE, CORRECT YOUR PROBLEM NUMBER. **		
				8320	LA 1,=CL82, **INITIALIZED ROUTINE SPECIFIED IS NOT AVAILABX		
				8321	LA 0(2,1),=X,4F02,		
				8322	LA 15,=A(SOSPRINT)		
				8323	LA 14,15		
				8324	GPRT 82,4F,02, **STUDENT HAS NOT INITIALIZED. CURRENT SVC NOT ACCEX		
				8325	ABLE, CORRECT YOUR PROBLEM NUMBER. **		
				8326	LA 1,=CL82, **INITIALIZED ROUTINE SPECIFIED IS NOT AVAILABX		
				8327	LA 0(2,1),=X,4F02,		
				8328	LA 15,=A(SOSPRINT)		
				8329	LA 14,15		
				8330	GPRT 82,4F,02, **STUDENT HAS NOT INITIALIZED. CURRENT SVC NOT ACCEX		
				8331	ABLE, CORRECT YOUR PROBLEM NUMBER. **		
				8332	LA 1,=CL82, **INITIALIZED ROUTINE SPECIFIED IS NOT AVAILABX		
				8333	LA 0(2,1),=X,4F02,		
				8334	LA 15,=A(SOSPRINT)		
				8335	LA 14,15		
				8336	GPRT 82,4F,02, **STUDENT HAS NOT INITIALIZED. CURRENT SVC NOT ACCEX		
				8337	ABLE, CORRECT YOUR PROBLEM NUMBER. **		
				8338	LA 1,=CL82, **INITIALIZED ROUTINE SPECIFIED IS NOT AVAILABX		
				8339	LA 0(2,1),=X,4F02,		
				8340	LA 15,=A(SOSPRINT)		
				8341	LA 14,15		
				8342	GPRT 82,4F,02, **STUDENT HAS NOT INITIALIZED. CURRENT SVC NOT ACCEX		
				8343	ABLE, CORRECT YOUR PROBLEM NUMBER. **		
				8344	LA 1,=CL82, **INITIALIZED ROUTINE SPECIFIED IS NOT AVAILABX		
				8345	LA 0(2,1),=X,4F02,		
				8346	LA 15,=A(SOSPRINT)		
				8347	LA 14,15		
				8348	GPRT 82,4F,02, **STUDENT HAS NOT INITIALIZED. CURRENT SVC NOT ACCEX		
				8349	ABLE, CORRECT YOUR PROBLEM NUMBER. **		
				8350	LA 1,=CL82, **INITIALIZED ROUTINE SPECIFIED IS NOT AVAILABX		
				8351	LA 0(2,1),=X,4F02,		
				8352	LA 15,=A(SOSPRINT)		
				8353	LA 14,15		
				8354	GPRT 82,4F,02, **STUDENT HAS NOT INITIALIZED. CURRENT SVC NOT ACCEX		
				8355	ABLE, CORRECT YOUR PROBLEM NUMBER. **		
				8356	LA 1,=CL82, **INITIALIZED ROUTINE SPECIFIED IS NOT AVAILABX		
				8357	LA 0(2,1),=X,4F02,		
				8358	LA 15,=A(SOSPRINT)		
				8359	LA 14,15		
				8360	GPRT 82,4F,02, **STUDENT HAS NOT INITIALIZED. CURRENT SVC NOT ACCEX		
				8361	ABLE, CORRECT YOUR PROBLEM NUMBER. **		
				8362	LA 1,=CL82, **INITIALIZED ROUTINE SPECIFIED IS NOT AVAILABX		
				8363	LA 0(2,1),=X,4F02,		
				8364	LA 15,=A(SOSPRINT)		
				8365	LA 14,15		
				8366	GPRT 82,4F,02, **STUDENT HAS NOT INITIALIZED. CURRENT SVC NOT ACCEX		
				8367	ABLE, CORRECT YOUR PROBLEM NUMBER. **		
				8368	LA 1,=CL82, **INITIALIZED ROUTINE SPECIFIED IS NOT AVAILABX		
				8369	LA 0(2,1),=X,4F02,		
				8370	LA 15,=A(SOSPRINT)		
				8371	LA 14,15		
				8372	GPRT 82,4F,02, **STUDENT HAS NOT INITIALIZED. CURRENT SVC NOT ACCEX		
				8373	ABLE, CORRECT YOUR PROBLEM NUMBER. **		
				8374	LA 1,=CL82, **INITIALIZED ROUTINE SPECIFIED IS NOT AVAILABX		
				8375	LA 0(2,1),=X,4F02,		
				8376	LA 15,=A(SOSPRINT)		
				8377	LA 14,15		
				8378	GPRT 82,4F,02, **STUDENT HAS NOT INITIALIZED. CURRENT SVC NOT ACCEX		
				8379	ABLE, CORRECT YOUR PROBLEM NUMBER. **		
				8380	LA 1,=CL82, **INITIALIZED ROUTINE SPECIFIED IS NOT AVAILABX		
				8381	LA 0(2,1),=X,4F02,		
				8382	LA 15,=A(SOSPRINT)		
				8383	LA 14,15		
				8384	GPRT 82,4F,02, **STUDENT HAS NOT INITIALIZED. CURRENT SVC NOT ACCEX		
				8385	ABLE, CORRECT YOUR PROBLEM NUMBER. **		
				8386	LA 1,=CL82, **INITIALIZED ROUTINE SPECIFIED IS NOT AVAILABX		
				8387	LA 0(2,1),=X,4F02,		
				8388	LA 15,=A(SOSPRINT)		
				8389	LA 14,15		
				8390	GPRT 82,4F,02, **STUDENT HAS NOT INITIALIZED. CURRENT SVC NOT ACCEX		
				8391	ABLE, CORRECT YOUR PROBLEM NUMBER. **		
				8392	LA 1,=CL82, **INITIALIZED ROUTINE SPECIFIED IS NOT AVAILABX		
				8393	LA 0(2,1),=X,4F02,		
				8394	LA 15,=A(SOSPRINT)		
				8395	LA 14,15		
				8396	GPRT 82,4F,02, **STUDENT HAS NOT INITIALIZED. CURRENT SVC NOT ACCEX		
				8397	ABLE, CORRECT YOUR PROBLEM NUMBER. **		
				8398	LA 1,=CL82, **INITIALIZED ROUTINE SPECIFIED IS NOT AVAILABX		
				8399	LA 0(2,1),=X,4F02,		
				8400	LA 15,=A(SOSPRINT)		
				8401	LA 14,15		
				8402	GPRT 82,4F,02, **STUDENT HAS NOT INITIALIZED. CURRENT SVC NOT ACCEX		
				8403	ABLE, CORRECT YOUR PROBLEM NUMBER. **		
				8404	LA 1,=CL82, **INITIALIZED ROUTINE SPECIFIED IS NOT AVAILABX		
				8405	LA 0(2,1),=X,4F02,		
				8406	LA 15,=A(SOSPRINT)		
				8407	LA 14,15		
				8408	GPRT 82,4F,02, **STUDENT HAS NOT INITIALIZED. CURRENT SVC NOT ACCEX		
				8409	ABLE, CORRECT YOUR PROBLEM NUMBER. **		
				8410	LA 1,=CL82, **INITIALIZED ROUTINE SPECIFIED IS NOT AVAILABX		
				8411	LA 0(2,1),=X,4F02,		
				8412	LA 15,=A(SOSPRINT)		
				8413	LA 14,15		
				8414	GPRT 82,4F,02, **STUDENT HAS NOT INITIALIZED. CURRENT SVC NOT ACCEX		
				8415	ABLE, CORRECT YOUR PROBLEM NUMBER. **		
				8416	LA 1,=CL82, **INITIALIZED ROUTINE SPECIFIED IS NOT AVAILABX		
				8417	LA 0(2,1),=X,4F02,		
				8418	LA 15,=A(SOSPRINT)		
				8419	LA 14,15		
				8420	GPRT 82,4F,02, **STUDENT HAS NOT INITIALIZED. CURRENT SVC NOT ACCEX		
				8421	ABLE, CORRECT YOUR PROBLEM NUMBER. **		
				8422	LA 1,=CL82, **INITIALIZED ROUTINE SPECIFIED IS NOT AVAILABX		
				8423	LA 0(2,1),=X,4F02,		
				8424	LA 15,=A(SOSPRINT)		
				8425	LA 14,15		
				8426	GPRT 82,4F,02, **STUDENT HAS NOT INITIALIZED. CURRENT SVC NOT ACCEX		
				8427	ABLE, CORRECT YOUR PROBLEM NUMBER. **		
				8428	LA 1,=CL82, **INITIALIZED ROUTINE SPECIFIED IS NOT AVAILABX		
				8429			

LCC	OBJECT CODE	ADDP1 ADDR2	STMT	SOURCE STATEMENT
017220	4110 C9F6	17856	9408 NODATA1 GPRT	95,52,02,1 **FIRST INTERMEDIATE ANSWER NOT CORRECT. DATA
017230	C201 1000	17874	9409+NCODATA1 LA	A NOT SUBSTITUTED. ROUTINE CONTINUED.**
017236	58E0 C738	17876	9410+ MVC	1,=CL86, **FIRST INTERMEDIATE ANSWER NOT CORRECT. DATA X
01723A	C5EE	17576	9411+ L	NOT SUBSTITUTED. ROUTINE CONTINUED.**
01723C	47F0 C3C5		9412+ BALR	0(2,1)=X(5202)
			9413+ R	15,=A(SOSPRINT)
				14,1F
				OUT
017240	45E0 C432	175D2	9415 INTANS2	14,GETANS
017244	C503 9004	17784	9416 CLC	ANS3,STUANS
01724A	4770 C0C2	17262	9417 RNE	FRR3
01724E	4110 C6E8	17888	9418 GPRT	40,25,01, **SECOND INTERMEDIATE ANSWER CORRECT**
017252	C201 1000	17976	9419+ LA	1,=CL40, **SECOND INTERMEDIATE ANSWER CORRECT**
017258	58E0 C738	178D8	9420+ MVC	0(2,1)=X(2501)
01725C	C5EE	175A0	9421+ L	15,=A(SOSPRINT)
01725E	47F0 C400		9422+ BALR	14,15
			9423+ R	MACHGRD
017262	5108 4005	00C05	9425 FRR3	GC8+1,X'09'
017266	4780 C0C8	17278	9426 TM	FRR31
01726A	4110 90A8	000A8	9427 RE	1,IMESS2
01726E	1211 C0D8	17278	9428 LA	1,1
017270	4780 C0D8	0225E	9429 BE	FRR31
017274	45E0 0158		9430 BAL	14,SOSPRINT
017278	5104 4005	00005	9432 FRR31	GC8+1,X'04'
01727C	4780 C0C8	1729A	9433 RE	NCODATA2
			9434 GPRT	R2,4F,01, **SECOND INTERMEDIATE ANSWER NOT CORRECT. DAX
017280	4110 C708	17978	9435+ LA	TA SUBSTITUTED. ROUTINE CONTINUED.**
017284	C201 1000	179CA	9436+ MVC	1,=CL82, **SECOND INTERMEDIATE ANSWER NOT CORRECT. DATA
01728A	58E0 C738	178D8	9437+ L	0(2,1)=X(4F01)
01728E	C5EE	17576	9438+ BALR	15,=A(SOSPRINT)
017290	6000 9004	00004	9439+ MVC	14,15
017296	47F0 C3C6		9440+ R	0(4,6),ANS2
				OUT
01729A	4110 C82C	179CC	9442 NODATA2 GPRT	96,53,02,1 **SECOND INTERMEDIATE ANSWER NOT CORRECT. DAX
01729F	C201 1000	17A22	9443+NCODATA2 LA	TA NOT SUBSTITUTED. ROUTINE CONTINUED.**
0172A4	58E0 C738	178D8	9444+ MVC	1,=CL86, **SECOND INTERMEDIATE ANSWER NOT CORRECT. DATA
0172A8	C5EE	17576	9445+ L	NOT SUBSTITUTED. ROUTINE CONTINUED.**
0172AA	47F0 C3C6		9446+ BALR	0(2,1)=X(5302)
			9447+ R	15,=A(SOSPRINT)
				14,15
				OUT
0172AE	45E0 C432	175D2	9449 INTANS3	14,GETANS
0172B2	0503 9008	17784	9450 CLC	ANS3,STUANS
0172B8	4770 C130	172D0	9451 RNE	FRR4
0172BC	4110 CA48	179F8	9452 GPRT	30,24,01, **THIRD INTERMEDIATE ANSWER CORRECT**
0172C6	C201 1000	1797C	9453+ LA	1,=CL39, **THIRD INTERMEDIATE ANSWER CORRECT**
0172CA	58E0 C738	178D8	9454+ MVC	0(2,1)=X(2401)
0172CC	47F0 C400	175A0	9455+ L	15,=A(SOSPRINT)
			9456+ BALR	14,15
			9457+ R	MACHGRD
0172D0	5109 4005	00005	9459 FPR4	GC8+1,X'08'
				TM

LOC	PROJECT CODE	ADDR1	ADDR2	STMT	SOURCE	STATEMENT	IF NOT, SEE WHAT INSTRUCTOR WANTS NEXT OBTAIN ADDRESS OF INSTRUCTOR'S MESSAGE IS MESSAGE DEFINED? IF NOT, NOOP PRINT INSTRUCTOR'S MESSAGE TO STUDENT
0172D4	478C C146		172EF	9450	RE	ERR41	
0172D8	411C 912E		0012E	9451	LA	1,1,IMFSS3	
0172D8	411C 912E		0012E	9452	LTR	1,1,IMFSS3	
0172D8	411C 912E		0012E	9453	RE	1,1,IMFSS3	
0172E2	45EC C158		0225E	9454	RA	14,5QSPRINT	
0172E6	5104 40C5	00005		9456	TM	GCR+1,X'04'	DOES INSTRUCTOR WANT DATA SUBSTITUTED?
0172EA	4780 C168		1730P	9457	RE	NODATA3	IF NOT, BRANCH
0172E2	4110 CA72		17C12	9459+	LA	1,4E,01'	**THIRD INTERMEDIATE ANSWER NOT CORRECT. DATA X
0172F2	4201 1000	C702	00000	9470+	MVC	A SUBSTITUTED	ROUTINE CONTINUED.**
0172F8	45FC C738		17972	9471+	L	1,4E,01'	**THIRD INTERMEDIATE ANSWER NOT CORRECT. DATA X
0172FC	45FC C738		17972	9472+	RA	1,4E,01'	**THIRD INTERMEDIATE ANSWER NOT CORRECT. DATA X
0172FE	4203 6000	9009	00000	9473	MVC	012,1)=X'2F01'	ROUTINE CONTINUED.**
017304	47E0 C3D6		17576	9474	R	014,6)=ANS3	GIVE THE STUDENT THE CORRECT ANSWER LET'S GO SEE WHAT THE STUDENT IS DOING
017308	4110 CAC3		17C63	9476	GPRT	85,52,02,1	**THIRD INTERMEDIATE ANSWER NOT CORRECT. DATA X
01730C	4201 1000	C704	00000	9477+	LA	A NOT SUBSTITUTED	ROUTINE CONTINUED.**
017312	45FC C738		17972	9478+	MVC	1,4E,01'	**THIRD INTERMEDIATE ANSWER NOT CORRECT. DATA X
017316	45FC C738		17972	9479+	L	012,1)=X'2F01'	ROUTINE CONTINUED.**
017318	47E0 C3D6		17576	9481	RA	15,6)=A(SOSPRINT)	
01731C	45E0 C432		175D2	9483	RA	14,GETANS	OBTAIN THE STUDENT'S ANSWER
017320	45E0 C432		175D2	9484	CLC	ANS3,STUANS	IS THE STUDENT'S ANSWER CORRECT?
01732E	4770 C19E		1733E	9485	BNE	ERR5	IF WRONG, ERROR MSG AND SUBSTITUTE
01732A	4110 C710		178RC	9487+	LA	40,25,01'	**FOURTH INTERMEDIATE ANSWER CORRECT**
01732E	4201 1000	C706	00000	9488+	MVC	1,4E,01'	**FOURTH INTERMEDIATE ANSWER CORRECT**
017334	45FC C738		1780P	9489+	L	012,1)=X'2F01'	ROUTINE CONTINUED.**
017338	45FC C738		1780P	9490+	L	15,6)=A(SOSPRINT)	
01733A	47FC C400		175AC	9491	RA	14,15	BRANCH TO TOTAL CREDIT FOR THIS ANSWER
01733E	5108 4005	00005		9493	TM	GCR+1,X'04'	DOES INSTRUCTOR HAVE ANYTHING TO SAY?
017342	4780 C184		17354	9494	RE	ERR51	IF NOT, SEE WHAT INSTRUCTOR WANTS NEXT
017346	4110 9184		00184	9495	LA	1,1,IMFSS4	OBTAIN ADDRESS OF INSTRUCTOR'S MESSAGE
017348	4211 C184		17354	9496	LTR	1,1,IMFSS4	IS MESSAGE DEFINED?
01734C	4780 C184		0225E	9497	RE	ERR51	IF NOT, NOOP
017350	45E0 D158		0225E	9498	RA	14,5QSPRINT	PRINT INSTRUCTOR'S MESSAGE TO STUDENT
017354	5104 40C5	00005		9501	TM	GCR+1,X'04'	DOES INSTRUCTOR WANT DATA SUBSTITUTED?
017358	4780 C1D6		17376	9502	RE	NODATA4	IF NOT, BRANCH
01735C	4110 C884		17A24	9503+	LA	1,4E,01'	**FOURTH INTERMEDIATE ANSWER NOT CORRECT. DATA X
017360	4201 1000	C82A	00000	9504+	MVC	A SUBSTITUTED	ROUTINE CONTINUED.**
017366	45FC C738		178D8	9505+	L	1,4E,01'	**FOURTH INTERMEDIATE ANSWER NOT CORRECT. DATA X
01736C	45FC C738		178D8	9506+	L	15,6)=A(SOSPRINT)	ROUTINE CONTINUED.**
017372	47E0 C3D6		17576	9507	RA	14,15	GIVE THE STUDENT THE CORRECT ANSWER LET'S GO SEE WHAT THE STUDENT IS DOING
017372	47E0 C3D6		17576	9508	B	014,6)=ANS4	
017372	47E0 C3D6		17576	9510	GPRT	86,53,02,1	**FOURTH INTERMEDIATE ANSWER NOT CORRECT. DATA X
017372	47E0 C3D6		17576	9511+	LA	A NOT SUBSTITUTED	ROUTINE CONTINUED.**
017372	47E0 C3D6		17576	9511+	LA	1,4E,01'	**FOURTH INTERMEDIATE ANSWER NOT CORRECT. DATA X

LOC	PROJECT CODE	ADDRESS	STMT	SOURCE	STATEMENT	STUDENT OPERATING SYSTEM	*** GRADER ***
01755F	C5EF		8740+		RALP 14,15		
017600	47E0	17576	8741		CUT		
017604	4110	C664	8742	CERR4	GOPT 45,24,01, **INCORRECT NO. OF INT. ANSWERS SPECIFIED**		
017608	C201	1000	8743+	CERR4	LA 1=C145, **INCORRECT NO. OF INT. ANSWERS SPECIFIED**		
01760F	84FC	C738	8744+		MVC 02,11)=X(2A01,		
017612	05EF		8745+		RALP 15,=A(SOSPEINT)		
017614	47FC	C306	8747		CUT		
017618			8748	GMW1	GMWSS 1132, \$\$\$, (GR1,52,4),C=2		TOTAL MACHINE GRADE=X
01761A			8749+		CRG *+2		
01761B			8750+	GRM1M	OC1132		
01761P	P3		8751+	GRM1	*-2		
01761S	C2		8752+		AL1(131)		
01761A	404C404040404040C		8753+		OC XLI12,		***END OF GRADER RUN: TOTAL MACHINE GRADEX
01764C			8754+		OC XLI132,		
01764C			8755+		= \$\$\$,		
01769F			8756+	GP1	GRM1M+52-1		
01769F			8757+		OC14		
0176AC			8758	GRM2	GMWSS 1132, ##\$, (GR2,47,4),C=2		***ACUMULATED INTERMEDATE ANSWER CREDIT= X
0176AC			8759+		CRG *+2		
0176AC			8760+	GRM2M	OC1132		
01769F	P3		8761+		*-2		
01769F	C2		8762+	GRM2	AL1(131)		
01769F			8763+		OC XLI12,		
0176AC	404C404040404040C		8764+		OC XLI132,		***ACUMULATED INTERMEDATE ANSWER CREDIT= X
0176CE			8765+		##\$,		
017724			8766+	GR2	GRM2M+47-1		
017724			8767+		OC14		
017726			8768	GRM3	GMWSS 1132, WITH BONUS POINTS= \$\$\$, (GR3,70,4),C=2		TOTAL MACHINE GRADE X
017726			8769+		CRG *+2		
017726			8770+	GRM3M	OC1132		
017724	P3		8771+		*-2		
017724	C2		8772+	GRM3	AL1(131)		
017725			8773+		OC XLI12,		
017726	404C404040404040C		8774+		OC XLI132,		***ADDITIONAL CREDIT: TOTAL MACHINE GRADEX
01776A			8775+		WITH BONUS POINTS= \$\$\$,		
01776A			8776+	GP3	GRM3M+70-1		
0177AA			8777+		OC14		
0177AA	C00C		8779	SUM	CRG		
0177AC	C0000000		8780	SUM1	DC		
0177AC	C0000000		8781	STUANS	DC		
0177AC			8782	WCRKWRN	DC		
0177AC	C0000000		8783	CREDIT	DC		
0177AC	C0000000		8784	ICREDIT	DC		
0177AC	C0000000		8785	XCREDIT	DC		
0177AC	P3		8786		DC		
0177AC	C2		8787		DC		
0177AC			8788	GEND	DC		
0177AC	F5C5C3C1C3E34CC		8789	CHAL1	DC		
017852	C905E2E3D9E4C3E3		8790	GHAT2	DC		
01787C	C200 C642 1000 177F2 C0CC		8791	GMVC	WVC		

NAVAL POSTGRADUATE SCHOOL	STUDENT OPERATING SYSTEM	*** GRADER ***
LOC	SUBJECT CODE	ADDR1 ADDR2
STMT	SOURCE STATEMENT	
9792	LTORG	
9793		
9794		
9795		
9796		
9797		
9798		
9799		
9800		
9801		
9802		
9803		
9804		
9805		
9806		
9807		
9808		
9809		
9810		
9811		
9812		
9813		
9814		
9815		
9816		
9817		
9818		
9819		
9820		
9821		
9822		
9823		
9824		
9825		
9826		
9827		
9828		
9829		
9830		
9831		
9832		
9833		
9834		
9835		

```

=CL40, **SECOND INTERMEDIATE ANSWER CORRECT**
=CL41, **FOURTH INTERMEDIATE ANSWER CORRECT**
=A(SOSPRINT)
=CL28, **FINAL ANSWER INCORRECT**
=F10,
=A(GEND+22)
=CL92, **YOUR PROGRAM REQUIRED MORE INSTRUCTIONS THAN TX
HE MEDIAN FOR PROJECTS IN THIS CATEGORY.**
=CL4,
=A(SAVE14)
=A(SAVE214)
=H1,
=X'2401,
=X'4E01,
=X'5202,
=X'2501,
=CL82, **SECOND INTERMEDIATE ANSWER NOT CORRECT. DATA SX
SUBSTITUTED. ROUTINE CONTINUED.**
=X'4E01, **SECOND INTERMEDIATE ANSWER NOT CORRECT. DATA NX
SUBSTITUTED. ROUTINE CONTINUED.**
=CL32, **FOURTH INTERMEDIATE ANSWER NOT CORRECT. DATA SX
SUBSTITUTED. ROUTINE CONTINUED.**
=CL84, **FOURTH INTERMEDIATE ANSWER NOT CORRECT. DATA NX
SUBSTITUTED. ROUTINE CONTINUED.**
=CL26, **FINAL ANSWER CORRECT**
=X'1702,
=X'1802,
=X'17C02,
=X'5902,
=X'0102,
=CL42, **SVC NOT FOLLOWED BY PROPER PARAMETER**
=X'2701,
=X'2A01,
=CL30,
=CL81, **FIRST INTERMEDIATE ANSWER CORRECT**
SUBSTITUTED. ROUTINE CONTINUED. ANSWER NOT CORRECT. DATA SUX
=CL85, **FIRST INTERMEDIATE ANSWER NOT CORRECT. DATA NCX
SUBSTITUTED. ROUTINE CONTINUED.**
=CL39, **THIRD INTERMEDIATE ANSWER CORRECT**
=CL81, **THIRD INTERMEDIATE ANSWER NOT CORRECT. DATA SUX
SUBSTITUTED. ROUTINE CONTINUED.**
=CL85, **THIRD INTERMEDIATE ANSWER NOT CORRECT. DATA NCX
SUBSTITUTED. ROUTINE CONTINUED.**
=CL39, **FIFTH INTERMEDIATE ANSWER CORRECT**
=CL81, **FIFTH INTERMEDIATE ANSWER NOT CORRECT. DATA SUX
SUBSTITUTED. ROUTINE CONTINUED.**
=CL85, **FIFTH INTERMEDIATE ANSWER NOT CORRECT. DATA NCX
SUBSTITUTED. ROUTINE CONTINUED.**
=CL127, **YOUR PROGRAM REQUIRED FEWER INSTRUCTIONS THANX
THE MEDIAN FOR PROJECTS IN THIS CATEGORY. ADDITIONAL CRX
EDIT IS GIVEN BELOW.**
=CL45, **INCORRECT NO. OF INT. ANSWERS SPECIFIED**

```


LDC	PROJECT CODE	ADDR1 ADDR2	STMT	SOURCE	STATEMENT
000239	C1		9955+	DC	XL1'01'
00023C	404C404C404C4C		9956+	DC	CL132'
0002CC			9957+	DC	CL132'
			9958+	IMESSF	GMFSS
			9959+	IMESSF	GMFSS
0002C2			9960+	IMESSFM	CL132
0002C2			9961+	IMESSFM	CL132
0002C0			9962+	IMESSF	CL132
0002CC	P3		9963+	IMESSF	CL132
0002C1	C1		9964+	DC	XL1'01'
0002C2	404C404C404C4C		9965+	DC	CL132'
000344				DC	CL132'

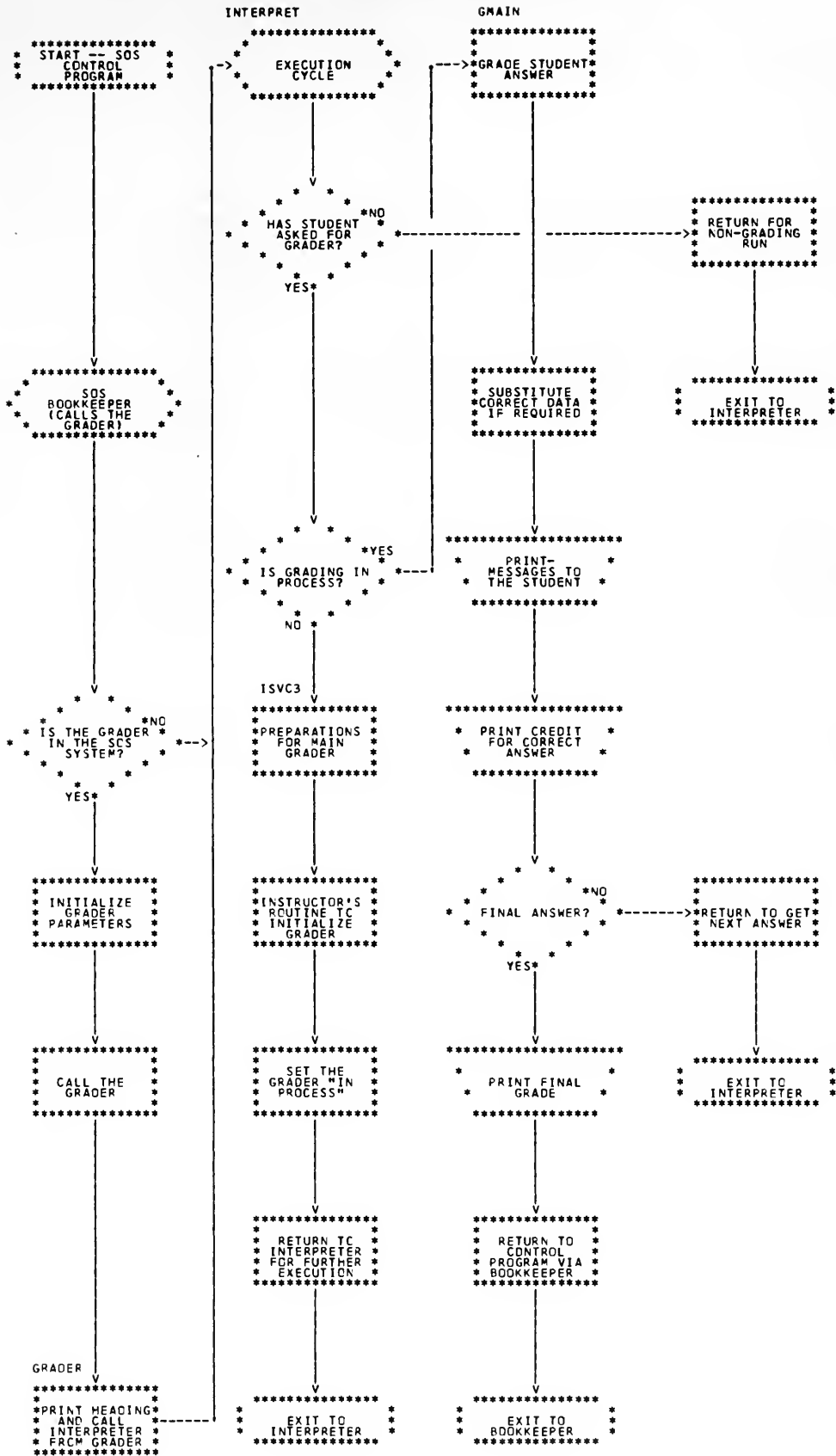
APPENDIX C

Flowchart Documentation of the Grader Program

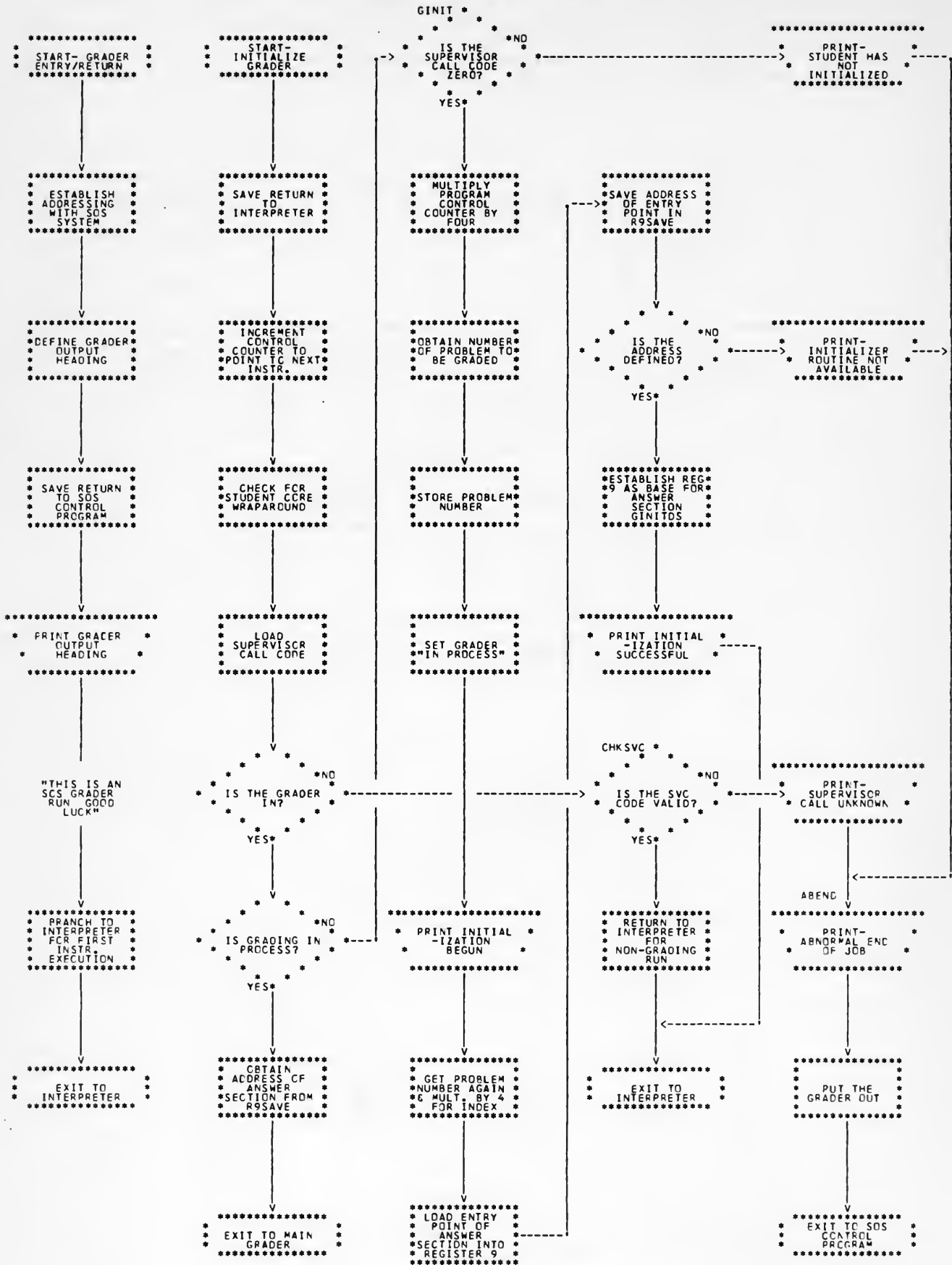
The flowcharts which document the various sections of the Grader Program are the following:

<u>Flowchart Number</u>	<u>Title</u>
1.	The Basic Grading Routine
2.	Grader/SOS Interface and Grader Initialization
3.	Intermediate Answer Grading Routine
4.	Final answer Grading Routine
5.	Routine for Accumulating Intermediate Answer Credit
6.	Routine to Obtain Student's Answer
7.	Translation and Printout of Machine Grade Routine
8.	Return Routines
9.	Total Machine Grade Routine (First Section)
10.	Total Machine Grade Routine (Second Section)
11.	Instructions Executed Routine
12.	Structure of General Communications Section

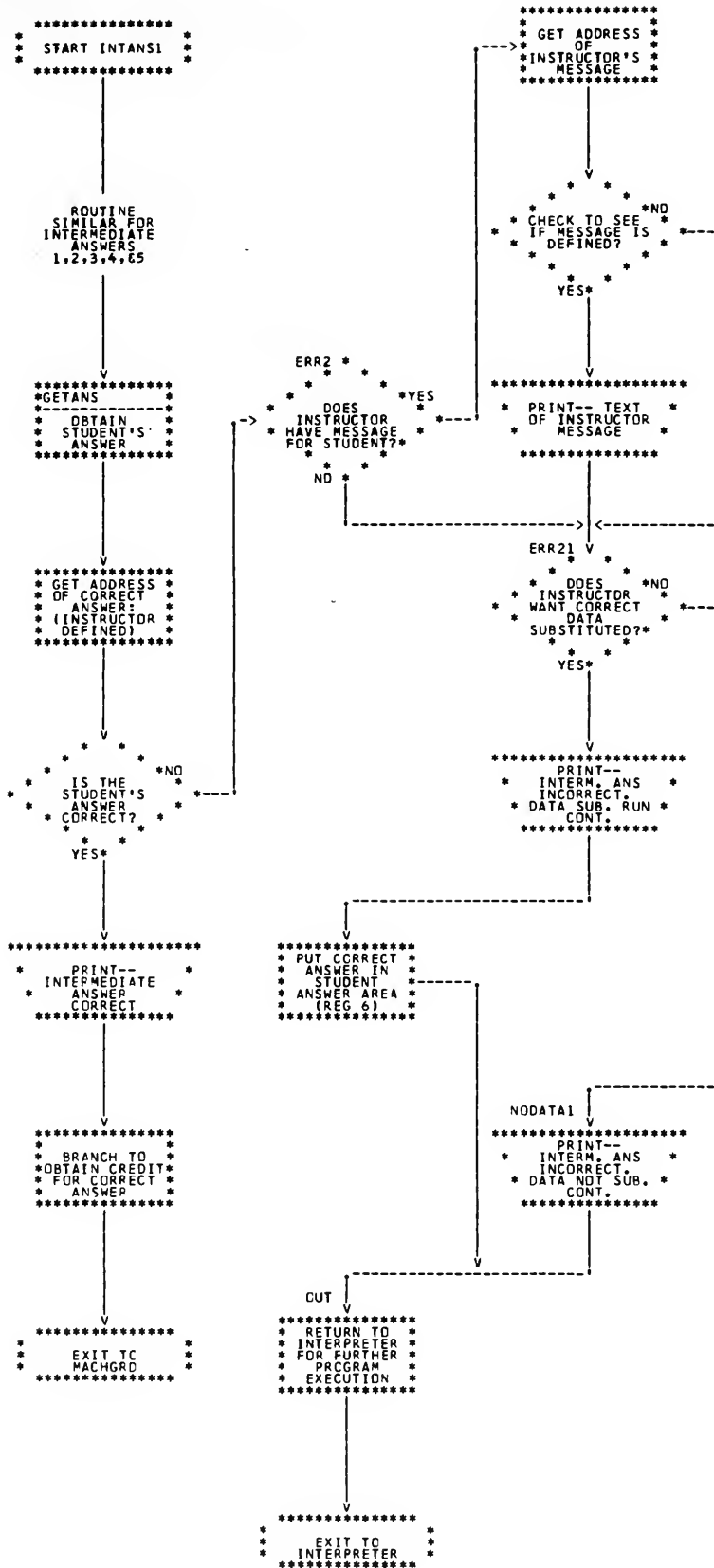
*** APPENDIX C -- CHART NUMBER 1 ***
 FLOW CHART OF THE BASIC GRADING ROUTINE



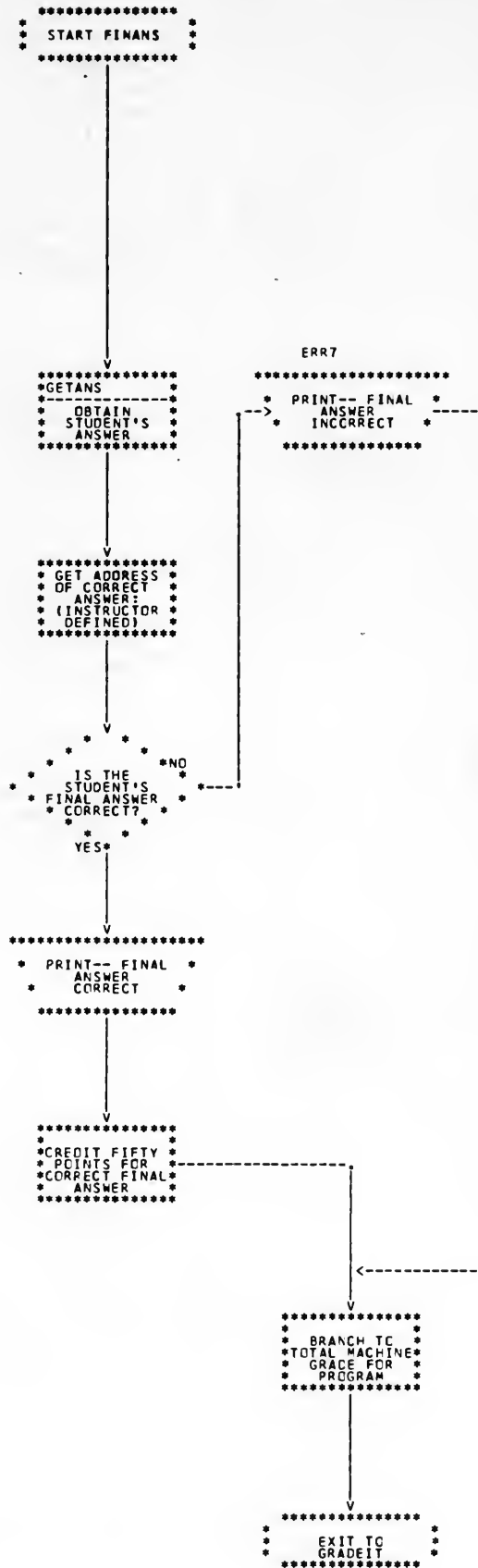
*** APPENDIX C -- CHART NUMBER 2 ***
 GRADER/SOS INTERFACE AND GRADER INITIALIZATION



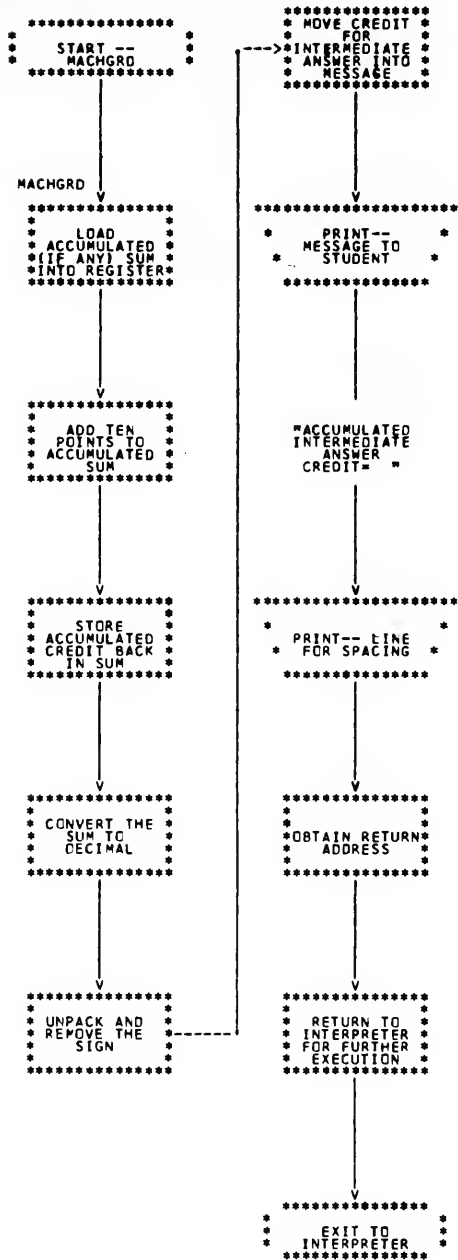
*** APPENDIX C -- CHART NUMBER 3 ***
INTERMEDIATE ANSWER GRADING ROUTINE



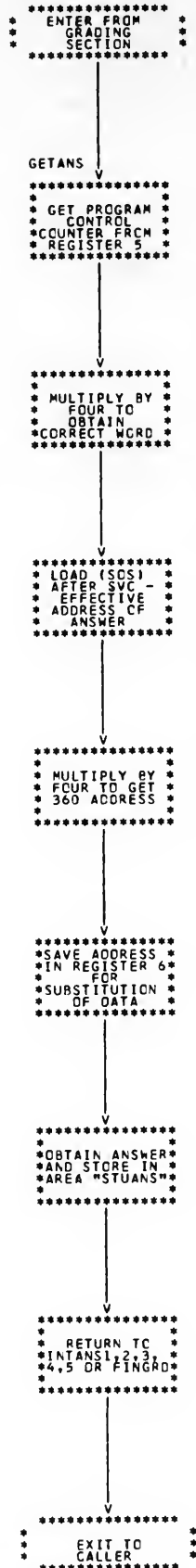
*** APPENDIX C -- CHART NUMBER 4 ***
 FINAL ANSWER GRADING ROUTINE



*** APPENDIX C -- CHART NUMBER 5 ***
ROUTINE FOR ACCUMULATING INTERMEDIATE ANSWER CREDIT

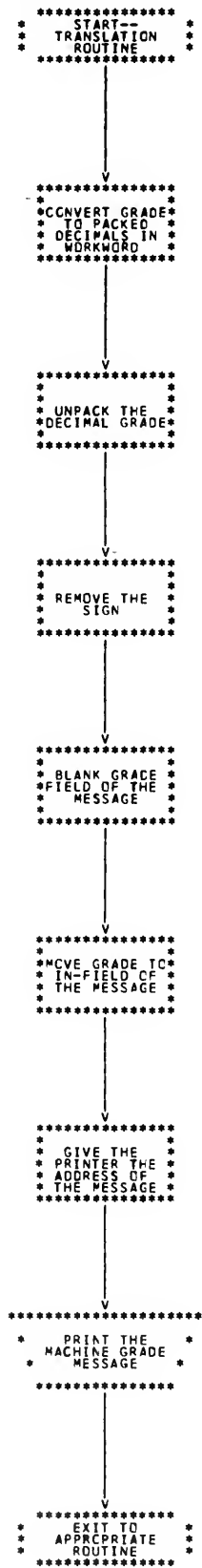


*** APPENDIX C -- CHART NUMBER 6 ***
ROUTINE TO OBTAIN STUDENT'S ANSWER



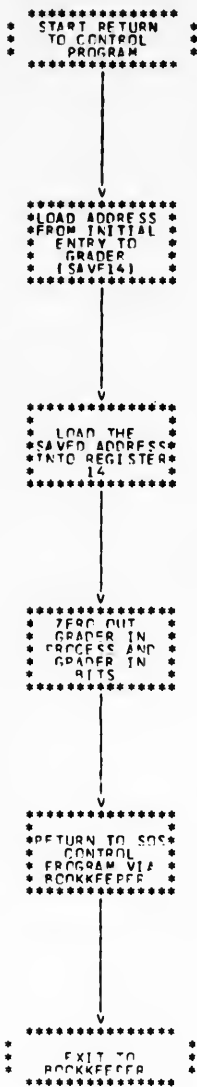
*** APPENDIX C -- CHART NUMBER 7 ***
 TRANSLATION AND PRINTOUT OF MACHINE GRADE ROUTINE

TRANS1,2,3

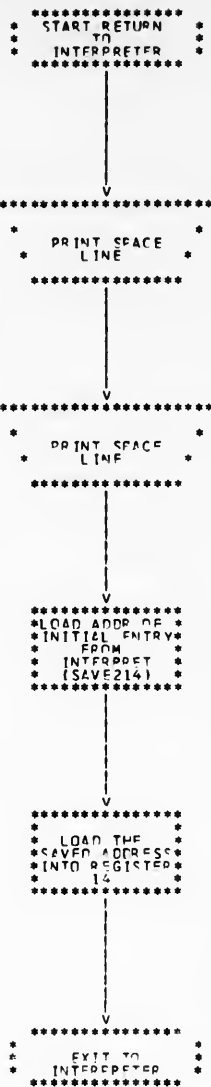


*** APPENDIX C -- CHART NUMBER 8 ***
RETURN ROUTINES

FINOUT1

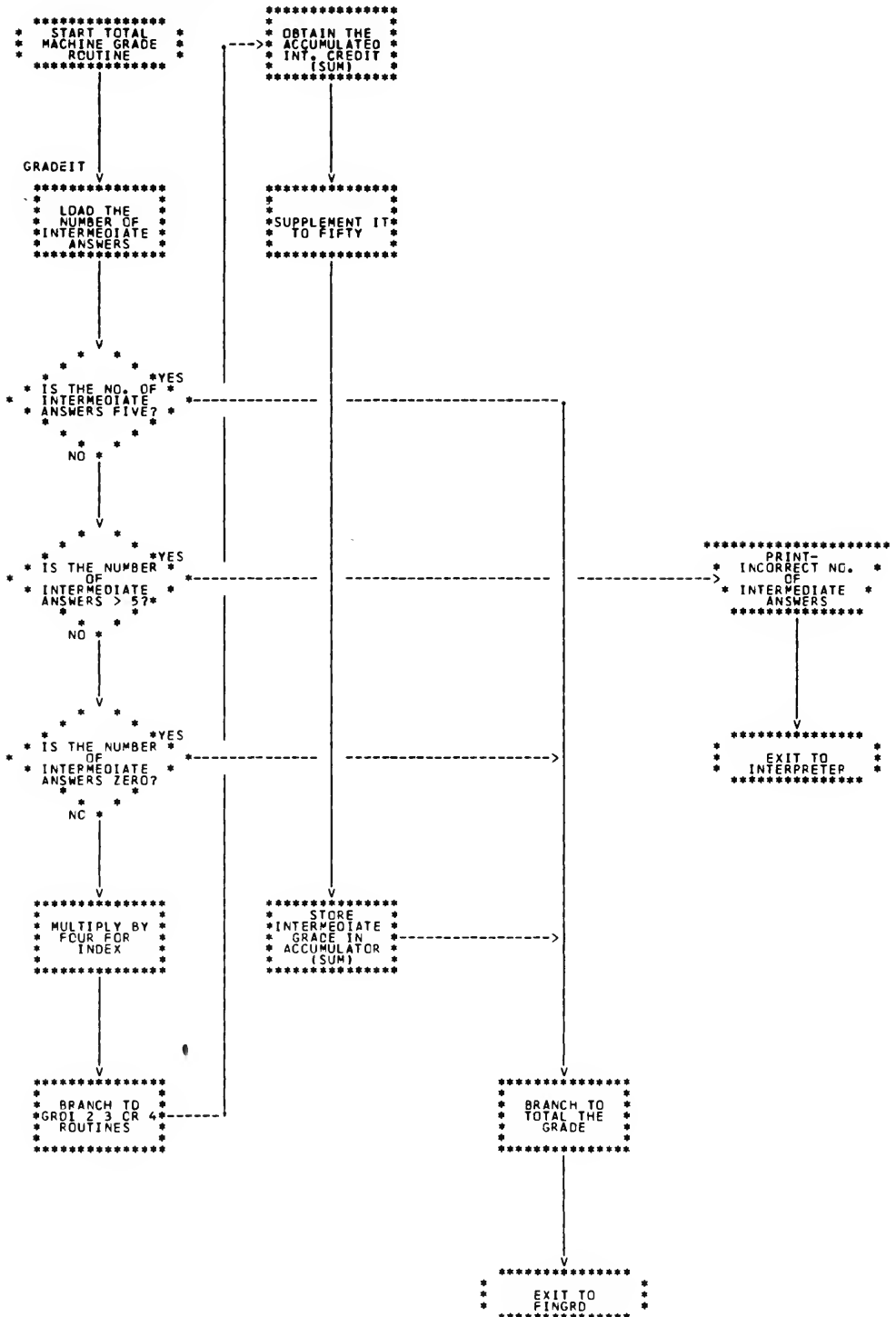


OUT

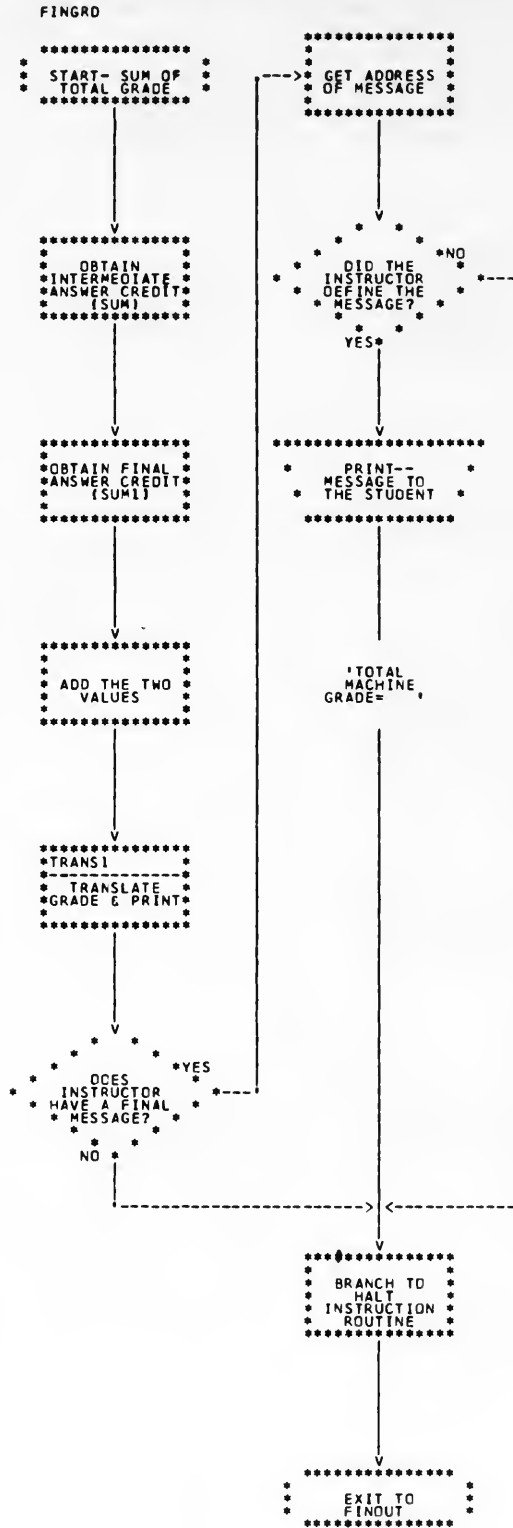


2426174

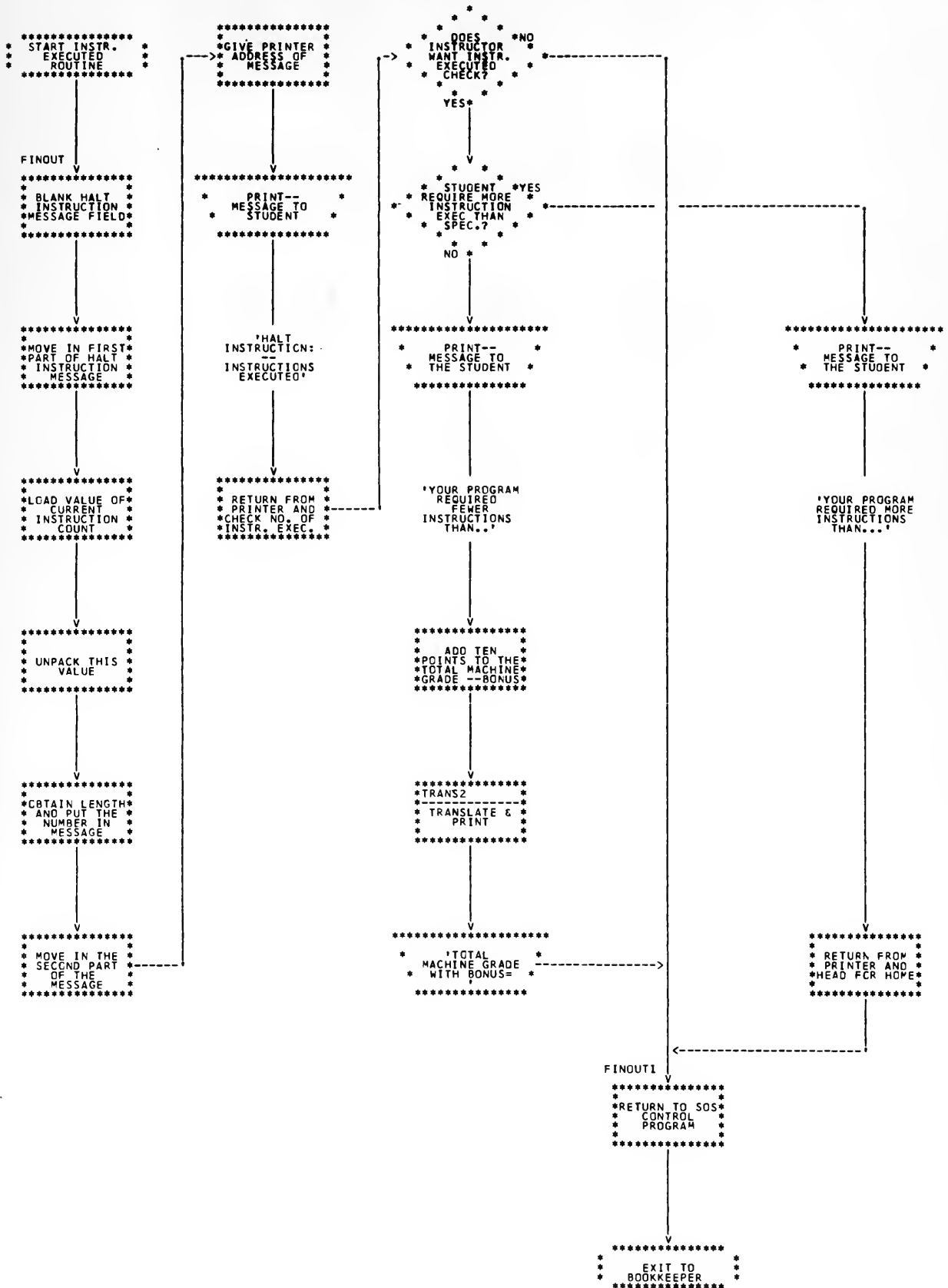
*** APPENDIX C -- CHART NUMBER 9 ***
TOTAL MACHINE GRADE ROUTINE (FIRST SECTION)



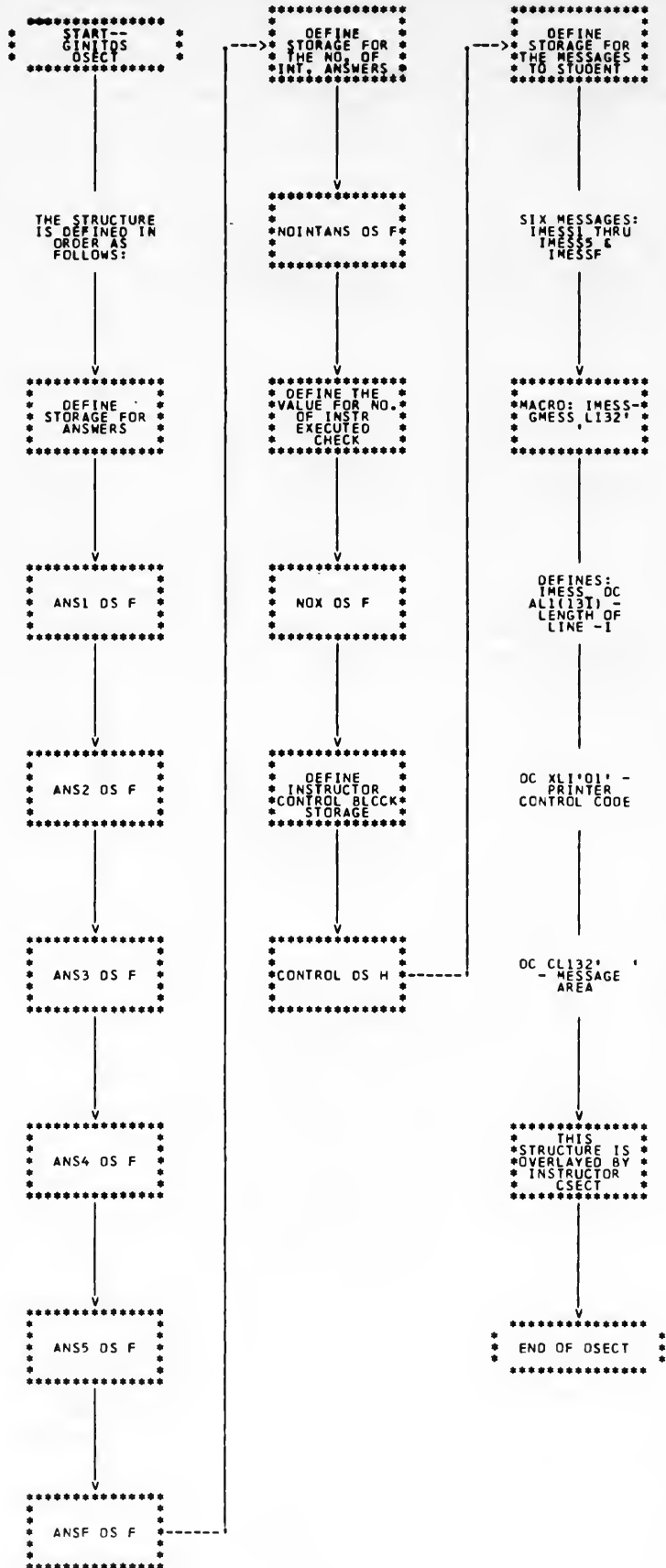
*** APPENDIX C -- CHART NUMBER 10 ***
TOTAL MACHINE GRADE ROUTINE (SECOND SECTION)



*** APPENDIX C -- CHART NUMBER-11 ***
INSTRUCTIONS EXECUTED ROUTINE



*** APPENDIX C -- CHART NUMBER 12 ***
STRUCTURE OF GENERAL COMMUNICATIONS SECTION



APPENDIX D

Examples of Graded Programs

An example of a program with the grader off is first shown.

Examples of graded programs are then given.

NAME: BACONR20 JCBNAME: EXAMPL20 NAVAL PCSTGRAD SCHOOL STUDENT OPERATING SYSTEM SCS/360 25 MAY, 1969
 BCKKEEPING PHASE MAP CF OPTIONS

```

CONFIGURATION--1,
CORE ZERGED--YES,
CUMP REQUESTED--YES,
GRADER--NO,
IMAX NUM EXECUTABLE INSTRUCTIONS--SCSASUM(3),
IMAX NUM PRINTED LINES--SOSASUM(2),
IMAX NUM TRACED INSTRUCTIONS--SOSASUM(1),
OUTPUT HEADER--TRACE,
STATUS--STUDENT,
TRACE INITIALLY--ON,
WARNINGS PRINTED--YES,

```

```

1SCSASUM=(500,1000,2000)
2SCSMAX=(200,1000,10000)

```

```

*****
* IF A MAN WILL BEGIN WITH CERTAINTIES *
* HE SHALL END IN DOUBTS; BUT IF HE *
* WILL BE CONTENT TO BEGIN WITH DOUBTS *
* HE SHALL END IN CERTAINTIES. *
* -- BACON *****

```

```

AC EDITING ERRORS OCCURRED DURING EDIT PHASE
PROGRAM EXAMPL20 WAS NOT MODIFIED
NEXT RUN IS NO. 2
EDIT PHASE COMPLETED---SUCCESSFULLY

```

LCCATION LCCOUNTER OBJECT CODE STATEMENT NUMBER

CARD IMAGE

 *** DEMONSTRATION OF GRADER ***

NOTE 1: EACH ANSWER DEPENDS
 ON PREVIOUS RESULTS.

NOTE 2: THE PROBLEM USES
 "IMMEDIATE" TYPE DATA.

NOTE 3: THE PROBLEM IS AN
 EXAMPLE OF GRADER INTER-
 ACTION ONLY AND IS NOT CON-
 SIDERED TO BE SUITABLE FOR
 A STUDENT TYPE PROBLEM.

FIRST ANSWER

THE FIRST TWO INSTRUCTIONS
 INITIALIZE THE GRADER FOR
 PROBLEM NO. 1. "TOFF" TURNS
 THE TRACE OFF. THE NUMBERS
 125 AND 24 ARE ADDED IN THE
 AND 50 IS SUBTRACTED IN THE
 EXECUTION OF THE SINGLE
 INSTRUCTION "AXAI".

010	C0300000	REGIN	SVC	3,0
011	00000000	DC	TOFF	0
015	A133007C	SXAI	3,125(3)	
016	A126C018	SXAI	4,24(4)	
017	A234FFCE	AXAI	3,50(4)	
018	60300064	ST	3,NUMBRS	

THE RESULT IS THEN PRINTED
 AND THE GRADER IS CALLED TO
 CHECK THE FIRST INTERMEDIATE
 ANSWER WHICH IS LOCATED AT
 THE EFFECTIVE ADDRESS OF
 "NUMBRS".

PUTC NUMBRS,NO=10
 RET
 SVC
 H

020 C03000C1
 021 00000064

NAME: RACONR20 JCBNAME: EXAMPL20 NAVAL PCSTGRAD SCHOOL STUDENT OPERATING SYSTEM SOS/360 29 MAY. 1969
SYMBCL TABLE
DEFINITION
BEGIN 00000010
EOP 00000063
LCCP1 00000023
NUMPRS 00000064
UNASSEMBLED STATEMENTS
NO COMPLETELY COMPLETED SUCCESSFULLY
***ASSEMBLY PHASE COMPLETED SUCCESSFULLY**
LCAD PHASE COMPLETED SUCCESSFULLY

NAME: BACNR20		JOBNAME: EXAMPL20		NAVAL PCSTGRAD		SCHOOL STUDENT		OPERATING SYSTEM		SOS/360		29 MAY, 1969			
CCNTRL	CURRENT	INST	REG	OLD REG	NEW REG	XREG	INDEX REG	IDADD	EFF	OLD WORD	NEW WORD				
CCOUNTER			NOS	CONTENTS	CONTENTS	NC	CONTENTS	LEVEL	ADD	CONTENTS	CONTENTS				
010	C03C0000								000	00000000					
012	9C0C0014														

014

99
2475
9900
16

2147483647
-2147483648

HALT INSTRUCTION: 104 INSTRUCTIONS EXECUTED
NCRPAL EXIT--END CF JCR

```

CCCONFIGURATION--1,
COPE ZEROED--YES,
CUMP REQUESTED--YES,
GRACER--YES,
I1MAX NUM EXECUTABLE INSTRUCTIONS--SOSASUM(3),
I1V1MAX NUM PRINTED LINES--SOSASUM(2),
I2MAX NUM TRACED INSTRUCTIONS--SOSASUM(1),
OUTPUT HEADER--TRACE,
STATUS--STUDENT,
TRACE INITIALLY--CN,
WARNINGS PRINTED--YES,

```

```
1SCSASUM=(500,1000,2000)
2SCSMAX=(200,1000,10000)
```

```
*****
** IF A MAN WILL BEGIN WITH CERTAINTIES **
** HE SHALL END IN DOUBTS; BUT IF HE **
** WILL BE CONTENT TO BEGIN WITH DOUBTS **
** HE SHALL END IN CERTAINTIES. **
** -- BACON **
*****
```

NO EDITING ERRORS OCCURRED DURING EDIT PHASE
PROGRAM EXAMPL21 WAS NOT MODIFIED
NEXT RUN IS NO. 2
EDIT PHASE COMPLETED---SUCCESSFULLY

LCCATION OBJECT STATEMENT
COUNTER CODE NUMBER

CARD IMAGE

```

*****
** DEMONSTRATION OF GRADER **
**
*****
**
NOTE 1: EACH ANSWER DEPENDS
ON PREVIOUS RESULTS.
**
NOTE 2: THE PROBLEM USES
"IMMEDIATE" TYPE DATA.
**
NOTE 3: THE PROBLEM IS AN
EXAMPLE OF GRADER INTER-
ACTION ONLY AND IS NOT CON-
SIDERED TO BE SUITABLE FOR
A STUDENT TYPE PROBLEM.
*****
** FIRST ANSWER **
*****
**
THE FIRST TWO INSTRUCTIONS
INITIALIZE THE GRADER FOR
PROBLEM NO. 1. "TOFF" TURNS
THE TRACE OFF. THE NUMBERS
125 AND 24 ARE THEN ADDED
AND 50 IS SUBTRACTED IN THE
EXECUTION OF THE SINGLE
INSTRUCTION "AXAI".
*****
BEGIN    SVC    3,0
         DC    0
         TOFF    0
         SXAI    3,125(1)
         SXAI    4,24(4)
         AXAI    3,-50(4)
         ST    3,NUMBR5
*****
**
THE RESULT IS THEN PRINTED
AND THE GRADER IS CALLED TO
CHECK THE FIRST INSTRUCTION
ANSWER WHICH IS LOCATED AT
THE EFFECTIVE ADDRESS OF
"NUMBR5".
*****
PUTD    NUMBR5,ND=10
RET    3,1
SVC    H
NUMBR5
*****

```

C10 C03C0000
C11 00000000
C15 A133C07D
C16 A144001B
C17 A234FFCE
C18 60300064

C20 CC300001
C21 00000064

NAME: BACNR21 JCBNAME: EXAMPL21 NAVAL PCSTGRAD SCHOOL STUDENT OPERATING SYSTEM SOS/360 25 MAY, 1969
SYMBCL TABLE
SYMBCL DEFINITION
BEGIN C0000010
ECF C0000063
LOCPI C0000023
NUMERS C0000064
ASSEMBLY PHASE COMPLETED SUCCESSFULLY
UNASSEMBLED STATEMENTS
COMPLETED SUCCESSFULLY
***LCAD PHASE COMPLETED SUCCESSFULLY**

NAME: BACNR21 JOBNAME: EXAMPL21 NAVAL PCSTGRAD SCHOOL STUDENT OPERATING SYSTEM SDS/360 29 MAY, 1969

THIS IS AN SCS GRADER RUN -- GOOD LUCK

PROB. 1 INITIALIZATION BEGUN

INITIALIZATION SUCCESSFUL

99
FIRST INTERMEDIATE ANSWER CORRECT
***ACCUMULATED INTERMEDIATE ANSWER CREDIT= 010 ***

2475
SECOND INTERMEDIATE ANSWER CORRECT
***ACCUMULATED INTERMEDIATE ANSWER CREDIT= 020 ***

9900
THIRD INTERMEDIATE ANSWER CORRECT
***ACCUMULATED INTERMEDIATE ANSWER CREDIT= 030 ***

16
FOURTH INTERMEDIATE ANSWER CORRECT
***ACCUMULATED INTERMEDIATE ANSWER CREDIT= 040 ***

2147483647
FIFTH INTERMEDIATE ANSWER CORRECT
***ACCUMULATED INTERMEDIATE ANSWER CREDIT= 050 ***

-2147483648
FINAL ANSWER CORRECT

\$\$\$END OF GRADER RUN: TOTAL MACHINE GRADE= 100 \$\$\$

INSTRUCTOR FINAL MESSAGE: THIS IS THE MINIMUM VALUE OF A DECIMAL CONSTANT

FALT INSTRUCTION: 103 INSTRUCTIONS EXECUTED

YOUR PROGRAM REQUIRED FEWER INSTRUCTIONS THAN THE MEDIAN FOR PROJECTS IN THIS CATEGORY. ADDITIONAL CREDIT IS GIVEN BELCW.

\$\$\$ADDITIONAL CREDIT: TOTAL MACHINE GRADE WITH BONUS POINTS= 110 \$\$\$

NAME: BACNR22 JOBNAME: EXAMPL22 NAVAL POSTGRAD SCHOOL STUDENT OPERATING SYSTEM SOS/360 29 MAY, 1969
 THIS IS AN SOS GRADER RUN -- GOOD LUCK
 PROB. 1 INITIALIZATION BEGUN
 INITIALIZATION SUCCESSFUL

100
 INSTRUCTOR MESSAGE NO. 1: USE SXAI TO LCAC CCNSTANTS INTO REGISTERS
 FIRST INTERMEDIATE ANSWER NOT CORRECT. DATA SUBSTITUTED. ROUTINE CONTINUED.

2475
 SECOND INTERMEDIATE ANSWER CORRECT
 ***ACCUMULATED INTERMEDIATE ANSWER CREDIT= 010 ***

9900
 THIRD INTERMEDIATE ANSWER CORRECT
 ***ACCUMULATED INTERMEDIATE ANSWER CREDIT= 020 ***

16
 FOURTH INTERMEDIATE ANSWER CORRECT
 ***ACCUMULATED INTERMEDIATE ANSWER CREDIT= 030 ***

2147483647
 FIFTH INTERMEDIATE ANSWER CORRECT
 ***ACCUMULATED INTERMEDIATE ANSWER CREDIT= 040 ***

-2147483648
 FINAL ANSWER CORRECT

\$\$\$END OF GRADER RUN: TOTAL MACHINE GRADE= 090 \$\$\$

INSTRUCTOR FINAL MESSAGE: THIS IS THE MINIMUM VALUE OF A DECIMAL CONSTANT
 HALT INSTRUCTION: 104 INSTRUCTIONS EXECUTED

YOUR PROGRAM REQUIRED FEWER INSTRUCTIONS THAN THE MEDIAN FOR PROJECTS IN THIS CATEGORY. ADDITIONAL CREDIT IS GIVEN BELCW.
 \$\$\$ADDITIONAL CREDIT: TOTAL MACHINE GRADE WITH BONUS POINTS= 100 \$\$\$

NAME: BACNR23 JCBNAME: EXAMPL23 NAVAL POSTGRAD SCHOOL STUDENT OPERATING SYSTEM SOS/360 29 MAY, 1969

THIS IS AN SCS GRADER RUN -- GOOD LUCK

PROB. 1 INITIALIZATION BEGUN

INITIALIZATION SUCCESSFUL

100

INSTRUCTOR MESSAGE NO. 1: USE SXAI TO LCAD CONSTANTS INTO REGISTERS

FIRST INTERMEDIATE ANSWER NOT CORRECT. DATA SUBSTITUTED. ROUTINE CONTINUED.

2476

INSTRUCTOR MESSAGE NO. 2: BCT R,LCC(X)

SECOND INTERMEDIATE ANSWER NOT CORRECT. DATA SUBSTITUTED. ROUTINE CONTINUED.

5900

THIRD INTERMEDIATE ANSWER CORRECT

***ACCUMULATED INTERMEDIATE ANSWER CREDIT= 010 ***

16

FOURTH INTERMEDIATE ANSWER CORRECT

***ACCUMULATED INTERMEDIATE ANSWER CREDIT= 020 ***

2147483647

FIFTH INTERMEDIATE ANSWER CORRECT

***ACCUMULATED INTERMEDIATE ANSWER CREDIT= 030 ***

-2147483648

FINAL ANSWER CORRECT

\$\$\$\$\$END OF GRADER RUN: TOTAL MACHINE GRADE= 080 \$\$\$\$

INSTRUCTOR FINAL MESSAGE: THIS IS THE MINIMUM VALUE OF A DECIMAL CONSTANT

FALT INSTRUCTION: 105 INSTRUCTIONS EXECUTED

YOUR PROGRAM REQUIRED FEWER INSTRUCTIONS THAN THE MEDIAN FOR PROJECTS IN THIS CATEGORY. ADDITIONAL CREDIT IS GIVEN RELCW.

\$\$\$\$\$ADDITIONAL CREDIT: TOTAL MACHINE GRADE WITH BONUS POINTS= 090 \$\$\$\$

NAME: BACONR24 JCBNAME: EXAMPL24 NAVAL POSTGRAD SCH70L STUDENT OPERATING SYSTEM SOS/360 29 MAY, 1969
THIS IS AN SOS GRADER RUN -- GOOD LUCK
PRCP. 1 INITIALIZATION BEGUN
INITIALIZATION SUCCESSFUL

99
FIRST INTERMEDIATE ANSWER CORRECT
***ACCUMULATED INTERMEDIATE ANSWER CREDIT= 010 ***

2475
SECCNC INTERMEDIATE ANSWER CORRECT
***ACCUMULATED INTERMEDIATE ANSWER CREDIT= 020 ***

9900
THIRC INTERMEDIATE ANSWER CORRECT
***ACCUMULATED INTERMEDIATE ANSWER CREDIT= 030 ***

16
FOURTH INTERMEDIATE ANSWER CORRECT
***ACCUMULATED INTERMEDIATE ANSWER CREDIT= 040 ***

2147483647
FIFTH INTERMEDIATE ANSWER CORRECT
***ACCUMULATED INTERMEDIATE ANSWER CREDIT= 050 ***

-2147483648
FINAL ANSWER CORRECT

\$\$\$END CF GRADER RUN: TOTAL MACHINE GRADE= 100 \$\$\$
INSTRUCTOR FINAL MESSAGE: THIS IS THE MINIMUM VALUE CF A DECIMAL CONSTANT
HALT INSTRUCTION: 1C9 INSTRUCTIONS EXECUTED
YOUR PROGRAM REQUIRED MORE INSTRUCTIONS THAN THE MEDIAN FOR PROJECTS IN THIS CATEGORY.

NAME: BACNR25 JOBNAME: EXAMPL25 NAVAL POSTGRAD SCHOOL STUDENT OPERATING SYSTEM SOS/360 29 MAY, 1969

THIS IS AN SCS GRADER RUN -- GOOD LUCK

PRCB. 1 INITIALIZATION BEGUN

INITIALIZATION SUCCESSFUL

¹⁰⁰
INSTRUCTOR MESSAGE NO. 1: USE SXAI TO LOAD CONSTANTS INTO REGISTERS
FIRST INTERMEDIATE ANSWER NOT CORRECT. DATA SUBSTITUTED. ROUTINE CONTINUED.

²⁴⁷⁶
INSTRUCTOR MESSAGE NO. 2: BCT R,LCC(X)
SECOND INTERMEDIATE ANSWER NOT CORRECT. DATA SUBSTITUTED. ROUTINE CONTINUED.

⁹⁹⁰¹
THE BEAUTY OF USING SHIFT INSTRUCTIONS INSTEAD OF MULTIPLICATION AND DIVISION BY 2**N IS THAT IT ONLY INVOLVES ONE REGISTER.
THIRD INTERMEDIATE ANSWER NOT CORRECT. DATA SUBSTITUTED. ROUTINE CONTINUED.

¹⁶
FOURTH INTERMEDIATE ANSWER CORRECT
***ACCUMULATED INTERMEDIATE ANSWER CREDIT= 010 ###

2147483647
FIFTH INTERMEDIATE ANSWER CORRECT
***ACCUMULATED INTERMEDIATE ANSWER CREDIT= 020 ###

-2147483648
FINAL ANSWER CORRECT

\$\$\$\$\$END OF GRADER RUN: TOTAL MACHINE GRADE= 070 \$\$\$\$

INSTRUCTOR FINAL MESSAGE: THIS IS THE MINIMUM VALUE OF A DECIMAL CONSTANT

ALT INSTRUCTION: 106 INSTRUCTIONS EXECUTED

YOUR PROGRAM REQUIRED MORE INSTRUCTIONS THAN THE MEDIAN FOR PROJECTS IN THIS CATEGORY.

NAME: BACNR26 JOBNAME: EXAMPL26 NAVAL PCSTGRAD SCHOOL STUDENT OPERATING SYSTEM SUS/360 25 MAY, 1969
 THIS IS AN SCS GRADER RUN -- GOOD LUCK
 PRCR. 1 INITIALIZATION BEGUN
 INITIALIZATION SUCCESSFUL

100
 INSTRUCTOR MESSAGE NO. 1: USE SXAI TO LOAD CONSTANTS INTO REGISTERS
 FIRST INTERMEDIATE ANSWER NOT CORRECT. DATA SUBSTITUTED. ROUTINE CONTINUED.

2476
 INSTRUCTOR MESSAGE NO. 2: BCT R,LCC(X)
 SECCND INTERMEDIATE ANSWER NOT CORRECT. DATA SUBSTITUTED. ROUTINE CONTINUED.

9901
 THE BEAUTY OF USING SHIFT INSTRUCTIONS INSTEAD CF MULTIPLICATION AND DIVISION BY 2**N IS THAT IT ONLY INVCLVESOME REGISTER.
 THIRD INTERMEDIATE ANSWER NOT CORRECT. DATA SUBSTITUTED. ROUTINE CONTINUED.

17
 IS THE INSTRUCTION # 16,LABEL VALID?
 FOURTH INTERMEDIATE ANSWER NOT CORRECT. DATA SUBSTITUTED. ROUTINE CONTINUED.

2147483647
 FIFTH INTERMEDIATE ANSWER CORRECT
 **ACCUMULATED INTERMEDIATE ANSWER CREDIT= 010 ###

-2147483648
 FINAL ANSWER CORRECT

\$\$\$END OF GRADER RUN: TOTAL MACHINE GRADE= 060 \$\$\$

INSTRUCTOR FINAL MESSAGE: THIS IS THE MINIMUM VALUE CF A DECIMAL CONSTANT

FALT INSTRUCTION: 1C7 INSTRUCTIONS EXECUTED

YOUR PROGRAM REQUIRED MORE INSTRUCTIONS THAN THE MEDIAN FOR PROJECTS IN THIS CATEGORY.

NAME: BACONR27 JOBNAME: EXAMPL27 NAVAL PCSTGRAD SCHOOL STUDENT OPERATING SYSTEM SOS/360 29 MAY, 1969
 THIS IS AN SCS GRADER RUN -- GOOD LUCK
 PROB. 1 INITIALIZATION BEGUN
 INITIALIZATION SUCCESSFUL

100
 INSTRUCTOR MESSAGE NO. 1: USE SXAI TO LOAD CONSTANTS INTO REGISTERS
 FIRST INTERMEDIATE ANSWER NOT CORRECT. DATA SUBSTITUTED. ROUTINE CONTINUED.

2476
 INSTRUCTOR MESSAGE NO. 2: BCT P,LOC(X)
 SECCND INTERMEDIATE ANSWER NOT CORRECT. DATA SUBSTITUTED. ROUTINE CONTINUED.

9901
 THE BEAUTY OF USING SHIFT INSTRUCTIONS INSTEAD CF MULTIPLICATION AND DIVISION BY 2**N IS THAT IT ONLY INVOLVES ONE REGISTER.
 THIRD INTERMEDIATE ANSWER NOT CORRECT. DATA SUBSTITUTED. ROUTINE CONTINUED.

17
 IS THE INSTRUCTION M 16,LABEL VALID?
 FOURTH INTERMEDIATE ANSWER NOT CORRECT. DATA SUBSTITUTED. ROUTINE CONTINUED.

2147483646
 WHY IS THE ABSOLUTE VALUE CF THIS ANSWER LESS THAN THE ABSOLUTE VALUE OF THE NEXT ANSWER?
 FIFTH INTERMEDIATE ANSWER NOT CORRECT. DATA SUBSTITUTED. ROUTINE CONTINUED.

-2147483648
 FINAL ANSWER CORRECT
 \$\$\$SEND OF GRADER RUN: TOTAL MACHINE GRADE= 050 \$\$\$
 INSTRUCTOR FINAL MESSAGE: THIS IS THE MINIMUM VALUE CF A DECIMAL CONSTANT
 HALT INSTRUCTION: 108 INSTRUCTIONS EXECUTED
 YOUR PROGRAM REQUIRED MORE INSTRUCTIONS THAN THE MEDIAN FOR PROJECTS IN THIS CATEGORY.

NAME: BACONR28 JCBNAME: EXAMPL28 NAVAL POSTGRAD SCHOOL STUDENT OPERATING SYSTEM SOS/360 29 MAY, 1969
 THIS IS AN SCS GRADER RUN -- GOOD LUCK
 PRCB. 1 INITIALIZATION REGUN
 INITIALIZATION SUCCESSFUL

¹⁰⁰
 INSTRUCTOR MESSAGE NO. 1: USE SXAI TO LOAD CCNstants INTO REGISTERS
 FIRST INTERMEDIATE ANSWER NOT CORRECT. DATA SUBSTITUTED. ROUTINE CONTINUED.

²⁴⁷⁶
 INSTRUCTOR MESSAGE NG. 2: BCT R,LOC(X)
 SECCND INTERMEDIATE ANSWER NOT CORRECT. DATA SUBSTITUTED. ROUTINE CONTINUED.

⁵⁹⁰¹
 THE BEAUTY OF USING SHIFT INSTRUCTIONS INSTEAD CF MULTIPLICATION AND DIVISION BY 2**N IS THAT IT ONLY INVCLVESCN REGISTER.
 THIRD INTERMEDIATE ANSWER NOT CORRECT. DATA SUBSTITUTED. ROUTINE CONTINUED.

¹⁷
 IS THE INSTRUCTION N 16,LABEL VALID?
 FOURTH INTERMEDIATE ANSWER NOT CORRECT. DATA SUBSTITUTED. ROUTINE CCNTINUED.

²¹⁴⁷⁴⁸³⁶⁴⁶
 WHY IS THE ABSOLUTE VALUE OF THIS ANSWER LESS THAN THE ABSOLUTE VALUE CF THE NEXT ANSWER?
 FIFTH INTERMEDIATE ANSWER NOT CORRECT. DATA SUBSTITUTED. ROUTINE CONTINUED.

⁻²¹⁴⁷⁴⁸³⁶⁴⁷
 FINAL ANSWER INCORRECT
 \$\$\$END OF GRADER PUN: TOTAL MACHINE GRADE= 050 \$\$\$
 INSTRUCTOR FINAL MESSAGE: THIS IS THE MINIMUM VALUE CF A DECIMAL CONSTANT
 FALT INSTRUCTION: ICS INSTPUCTIONS EXECUTED
 YOUR PROGRAM REQUIRED MORE INSTRUCTIONS THAN THE MEDIAN FOR PROJECTS IN THIS CATEGORY.

APPENDIX E

Example of Instructor's Answer Control Section.

*** APPENDIX E: INSTRUCTOR'S ANSWER CONTROL SECTION ***

LOC	OBJECT CODE	ADDR1	ADDR2	STMT	SOURCE STATEMENT
41	ENAME			41	MACRO
42	PROF			42	LCLA EN,EB1,EB2,EB3
43	LCLA			43	
44	*			44	
45	*			45	
46	*			46	
47	*			47	
48	*			48	
49	*			49	
50	*			50	
51	*			51	
52	*			52	
53	*			53	
54	*			54	
55	*			55	
56	*			56	
57	*			57	
58	*			58	
59	*			59	
60	*			60	
61	EB1			61	(N'ESYSLIST GT 3).ERROR1
62	EB2			62	AIF
63	EB3			63	SETA 1
64	EB3			64	SETA 1
65	EB3			65	AIF (N'ESYSLIST EQ 0).HERE
66	EB3			66	ANOP
67	EB3			67	SETA EN+1
68	EB3			68	(N'ESYSLIST(EN)) EQ 'NOMSG').SET1
69	EB3			69	(N'ESYSLIST(EN)) EQ 'NODATA').SET2
70	EB3			70	(N'ESYSLIST(EN)) EQ 'NOBONUS').SET3
71	EB3			71	(N'ESYSLIST(EN)) EQ 'NOBONUS').SET3
72	EB3			72	.HERE
73	EB3			73	AG0
74	EB3			74	ANOP
75	EB3			75	SETA 0
76	EB3			76	AG0 .LOOP
77	EB3			77	ANOP
78	EB3			78	SETA 0
79	EB3			79	AG0 .LOOP
80	EB3			80	ANOP
81	EB3			81	SETA 0
82	EB3			82	AG0 .LOOP
83	EB3			83	MNOTE *
84	EB3			84	HERE
85	EB3			85	CONTROL
86	EB3			86	DC X'FF'
87	EB3			87	DC R'1111EB1EB2EB3.1'
88	EB3			88	MEND

*** APPENDIX E: INSTRUCTOR'S ANSWER CONTROL SECTION ***

LOC	OBJECT CODE	ADDR1	ADDR2	STMT	SOURCE STATEMENT
90				90	EGNAME MACRO MSG
91				91	*****
92				92	*****
93				93	*****
94				94	*****
95				95	*****
96				96	*****
97				97	*****
98				98	*****
99				99	*****
100				100	*****
101				101	*****
102				102	*****
103				103	*****
104				104	*****
105				105	*****
106				106	*****
107				107	*****
108				108	*****
109				109	*****
110				110	*****
111				111	*****

*** APPENDIX E: INSTRUCTOR'S ANSWER CONTROL SECTION ***

LCC	OBJECT CODE	ADDR1	ADDR2	STMT	SOURCE STATEMENT
000000				114	GINIT
000001	00000063			115	PROFS
000002	00000063			116	ANS1
000003	00000063			117	ANS2
000004	00000063			118	ANS3
000005	00000063			119	ANS4
000006	00000063			120	ANS5
000007	00000063			121	ANS6
000008	00000063			122	NOINTANS
000009	00000063			123	NOX
000010	00000063			124	PROFS
000011	00000063			125	CONTROL
000012	00000063			126	IMESS1
000013	00000063			127	IMESS1
000014	00000063			128	IMESS1
000015	00000063			129	IMESS1
000016	00000063			130	IMESS1
000017	00000063			131	IMESS1
000018	00000063			132	IMESS1
000019	00000063			133	IMESS1
000020	00000063			134	IMESS2
000021	00000063			135	IMESS2
000022	00000063			136	IMESS2
000023	00000063			137	IMESS2
000024	00000063			138	IMESS2
000025	00000063			139	IMESS2
000026	00000063			140	IMESS2
000027	00000063			141	IMESS2
000028	00000063			142	IMESS2
000029	00000063			143	IMESS2
000030	00000063			144	IMESS3
000031	00000063			145	IMESS3
000032	00000063			146	IMESS3
000033	00000063			147	IMESS3
000034	00000063			148	IMESS3
000035	00000063			149	IMESS3
000036	00000063			150	IMESS4
000037	00000063			151	IMESS4
000038	00000063			152	IMESS4
000039	00000063			153	IMESS4
000040	00000063			154	IMESS4
000041	00000063			155	IMESS4
000042	00000063			156	IMESS4
000043	00000063			157	IMESS4
000044	00000063			158	IMESS4
000045	00000063			159	IMESS5
000046	00000063			160	IMESS5
000047	00000063			161	IMESS5
000048	00000063			162	IMESS5
000049	00000063			163	IMESS5
000050	00000063			164	IMESS5
000051	00000063			165	IMESS5

BIBLIOGRAPHY

1. Hollingsworth, Jack, "Automatic Graders for Programming Classes," Communications of the ACM, v. 3, p. 528-529, October 1960.
2. Forsythe, George E., and Wirth, Nicklaus, "Automatic Grading Programs," Communications of the ACM, v. 8, p. 275-278, May 1965.
3. Forsythe, George E., Automatic Machine Grading Systems, Proceedings A. C. M. National Conference, 1964, p. N1.4-1, A.C.M., 1964.
4. Berry, R. E., "Grader Programs," The Computer Journal, v. 9, p. 252-256, 1966.
5. Naur, P., "Automatic Grading of Students ALGOL Programming," B.I.T., v. 4, p. 177, 1964.
6. Temperley, J. F., and Smith, Barry W., "A Grading Procedure for PL/I Student Exercises," The Computer Journal, v. 10, p. 368-374, February 1968.
7. Hext, J. B., and Winings, J. W., "An Automatic Grading Scheme for Simple Programming Exercises," Communications of the ACM, v. 12, p. 272-275, May 1969.
8. Montalbano, Michael S., Expressing Program Logic, Data Processing Digest, v. 14, p. 1-16, September 1968.
9. Opler, Ascher, Is Assembly Language Programming Passe' ?, Data Processing Digest, v. 14, p. 1-15, October 1968.
10. Wile, David S., Munck, Robert G., and van Dam, Andries, The Brown University Student Operating System, Proceedings of 22nd National Conference, A.C.M., p. 427-439, Thompson Book Company, 1967.
11. Brown University Students in A. M. 101-102, SOS Brown University Student Operating System for IBM System/360 Model 50, Center for Computer and Information Sciences, Brown University, November, 1968.

INITIAL DISTRIBUTION LIST

	No Copies
1. Defense Documentation Center Cameron Station Alexandria, Virginia 22314	20
2. Library, Code 0212 Naval Postgraduate School Monterey, California 93940	2
3. Chief of Naval Operations (OP-91) Department of the Navy Washington, D.C. 20350	1
4. Prof. G. L. Barksdale, Jr. (thesis advisor) Department of Mathematics (Code 53Bv) Naval Postgraduate School Monterey, California 93940	1
5. Prof. Edward A. Singer Department of Mathematics (Code 53Sf) Naval Postgraduate School Monterey, California 93940	1
6. Prof. W. S. Brainerd Department of Mathematics (Code 53Bz) Naval Postgraduate School Monterey, California 93940	1
7. Prof. D. G. Williams Computing Center (Code 0211) Naval Postgraduate School Monterey, California 93940	1
8. Prof. Andries van Dam Brown University Providence, Rhode Island 02912	1
9. Richard M. Kogut Computer Laboratory Brown University Providence, Rhode Island 02912	1
10. LCDR Roger F. Bacon, USN (student) USS HALIBUT (SSN-587) FPO San Francisco 96601	1

Unclassified

Security Classification

DOCUMENT CONTROL DATA - R & D

(Security classification of title, body of abstract and indexing annotation must be entered when the overall report is classified)

1. ORIGINATING ACTIVITY (Corporate author) Naval Postgraduate School Monterey, California		2a. REPORT SECURITY CLASSIFICATION Unclassified	
		2b. GROUP	
3. REPORT TITLE An Investigation of An Automatic Machine Grading System for Assembly Language Instruction			
4. DESCRIPTIVE NOTES (Type of report and, inclusive dates) Master's Thesis; (June 1969)			
5. AUTHOR(S) (First name, middle initial, last name) Roger Francis Bacon			
6. REPORT DATE June 1969	7a. TOTAL NO. OF PAGES 105	7b. NO. OF REFS 11	
8a. CONTRACT OR GRANT NO.		9a. ORIGINATOR'S REPORT NUMBER(S)	
b. PROJECT NO.			
c.		9b. OTHER REPORT NO(S) (Any other numbers that may be assigned this report)	
d.			
10. DISTRIBUTION STATEMENT Distribution of this document is unlimited			
11. SUPPLEMENTARY NOTES		12. SPONSORING MILITARY ACTIVITY Naval Postgraduate School Monterey, California 93940	

13. ABSTRACT <p>Machine grading techniques are becoming of greater importance because of the increased number of students in programming classes. Characteristics and limitations of automatic machine grading systems are proposed. A grader for introductory assembly language programming courses was developed. The properties of this grader are discussed and an example of a graded program is given.</p>

KEY WORDS

LINK A

LINK B

LINK C

ROLE

WT

ROLE

WT

[illegible]

WT

Computer Grading Programs

Thesis
B115 Bacon
c.1 109295
An investigation of
an automatic machine
grading system for
assembly language in-
struction.

Thesis
B115 Bacon
c.1 109295
An investigation of
an automatic machine
grading system for
assembly language in-
struction.

thesB115

An investigation of an automatic machine



3 2768 000 99011 3

DUDLEY KNOX LIBRARY